REMARKS

This Amendment and Response is responsive to the office action mailed September 26, 2002. Applicant sincerely appreciates the Examiner's assistance in preparing this *pro se* application. Thank you for pointing out the necessary areas for correction and allowing applicant the opportunity to submit the missing drawings necessary to cover all of the claimed elements.

The Office Action has been carefully considered. Reconsideration of the specification, drawings, and claims as amended, and in view of the following remarks, is respectfully requested.

A. Summary of Examiner's Action and Applicant's Response

1. Drawing Objections

The drawings were objected to under 37 CFR 1.83(a). Additional drawing figures are submitted to show every feature of the invention specified in the claims. Please consider the attached proposed informal drawings.

2. Specification Objection

The specifications were objected to under 37 CFR 1.75(d)(1) and MPEP 608.01(o). for failing to provide proper antecedent basis for the claimed subject matter. The required amendments have been made to the specification to address this rejection.

2. Claim Objections

Claims 1 and 5 were objected to because of informalities. Claims 1, 5, *and 7* have been amended to correct obvious informalities.

2. Prior Art Rejections

Claims 1-3, 5-9, 11, 12, and 14-19 were rejected as unpatentable under 35 U.S.C 102(b) as being anticipated by Astle (5,552,832), Chaddha et al. (5, 621,660), Walsh (5,646,618), or Agarwal (5,812,788). Claims 4, 10, 13, 20 and 21 were rejected under 35 U.S.C. 103(a) as being obvious in view of Astle in view of one of Frederiksen (4,743,959), Anastassiou (4,545,385), or Pinder

(6,219,358 B1). Applicant respectfully traverses these rejections and requests reconsideration for the reasons discussed below.

B. Applicant's Terminology Sampling and Sub-sampling

In applicant's co-pending application serial number 09/470,566, applicant used two separate claim terms "sub-sampling pixels" and "sub-sampling frames". In this application the claim term "sub-sampling a number pixels bits" (claim 1(a)) is one characteristic of "sub-sampling pixels" and the claim term "image dimensions" (claim 2) , or "the width and the height of said image" (claim 9) relate to another distinct characteristic of "sub-sampling" from an original sample of pixels that make up a image or frame of a video stream.

As explained in the subject application (and in the 09/470,566 specification, including the 09/312,922 application, which are both now included by reference), when video is converted from analog to digital, the infinite analog signal must be "*sampled*" to create a finite number of digital values (page 3, Video Signals). Standard video is normally *sampled* at a frame rate (temporal resolution) of 30 frames per second (images per second), at a width (horizontal resolution) of 640 pixels per line, at a height (vertical resolution) of 480 pixels per line, and a pixel bit depth (color resolution or color depth) of 24 or 32 bits per pixels. The width and height combined together also are typically referred to in the art as dimensions (spatial resolution). When a subset of the original samples is taken, in any of these various resolutions, from the original sample set, the taking of the subset is referred to as "*sub-sampling*". Thus frame rate, image width and height, and pixel bit depth can all be independently *sub-sampled*. For example, if an original sample were taken at 30 frames per second, a sub-sampling of every third frame would result in a frame sub-sample of 10 frames per second. Further, for example, if an original sample were taken at 640 pixels per line, a sub-sampling of every fourth pixel would result in a pixel width of 160. Further, for example, if an original sample were taken at 480 pixels per column, a sub-sampling of every sixth pixel would result in a pixel height of 80. Further, for example, if an original sample were taken at 32 bits per pixel, a sub-sampling of 6 bits of each of the three color components would result in an 18 bit color depth.

Further, an image may be sub-sampled by selecting a portion of an image, as disclosed in co-pending application 09/433,978, filed 4 November 1999. When a portion of an image is sampled, the

subset has the same spatial resolution as the original but area of the sub-sample is less than the area of the original sample. For consistency, this sub-sampling could be referred to as area sub-sampling.

In the specification of the subject invention, Applicant described "sub-sampling a number of pixel bits" in reference to figures 2C through 2H, 3A, 3B, 4A (step 405), and 8B. The provisional application Serial Number 60/113,051 gave examples of potentially claimable pixel sub-sampling as follows:

13. When W is 8, the following value pairs for N and U are claimed: (2,6), (3,5), (4,4), (5,3), and (6,2).
14. When W is 16, the following value pairs for N and U are claimed: (2,14), (3,13), (4,12), (5,11), (6,10), (7, 9), (8, 8), (9, 7), (10, 6), (11, 5), (12, 4), (13, 3), and (14, 2).
15. When W is 32, all combinations of values pairs for N and U are claimed, where N+U equals 32 and N > 1 and U > 1.
16. When W is not a multiple of 8, all combinations of values pairs for N and U are claimed, where N+U equals W and N > 1 and U > 1.

Note that in all of these examples the number of ***sub-sampled*** pixel bits N is less than the original *sample* number of bits W because the number of unused bits U is always greater that 1. This will be discussed below in reference the claim 3 rejection based on Astle's original sample number of bits where W is 24.

Discussion of Cited Prior Art:

Astle, Chaddha, Walsh, Agarwal, Frederiksen, Anastassiou and Pinder

1. **Overview**

The Astle, Chaddha, Walsh and Agarwal, or Astle in combination of Frederiksen, Anastassiou, or Pinder, are the basis for all of the prior art rejections. These references will be dicussed below.

Anastassiou is the earliest patent (filed in 1983, issued in 1985) and is limited to the compression of low-resolution black and white text, or line art drawings using two bit planes. All of the other references, with the exception of Anastassiou, disclose rather complicated compression methods that employ some form of block based, lossy transform steps, followed by one or quantization steps, and entropy encoding steps. All of the references, including Anastassiou, differ significantly from the subject invention and teach away from the simple, fast, one-the-fly, one pass, clinically lossless method and apparatus of the present invention.

Applicant notes that Shimizu et al. (6,064,324) and Hartug et al. (5,309,232) were made of record but not relied upon.

2. Astle is a Complex, Block-based, Lossy Compression Method

Astle is a complex, block based, lossy compression method. Specifically, Astle sub-samples YUV signals, prefilters the signals, transforms the signals to obtain coefficients. Next the *coefficients*, not "*a number of pixels bits*", are run length encoded. After the *coefficent* run length encoding step, a variable length coding step is used to generate a bit stream. Astles YUV 4:1:1 subsampling (Fig 4, 404) across a plurality of pixels is different than Applicants "sub-sampling a number of pixel bits" from each single pixel. Astles "host processor and memory" (Fig 2, 202) is not the same as Applicant's "encoded data buffer". Each of the elements of Astle is very different than those of applicant's invention.

Further Astle discloses a rather complicated compression method that employs a form of block-based, lossy transform steps, followed by one or quantization steps, and entropy encoding steps. This reference differs significantly from the subject invention and teaches away from the simple, fast, one-the-fly, one pass, clinically lossless method and apparatus of the present invention.

3. Chaddha is a Complex, Block-based, Lossy Compression Method

Chaddha is a complex, block based, lossy compression method. Specifically, Chaddha is not designed to encode in real time but is a system in which video signals are preprocessed to create multiple video streams which are stored together on video disks; when the video is retrieved from the storage video disks, rate scaling is applied to select from one or more of the preprocessed streams. Chaddha encoding is so complex that the decompressor (Fig 2, 270) is required on the encoding side. Applicant's invention is much different.

Chaddha's decimate step (Fig 2, 210) is not "sub-sampling a number of *pixel bits*" from each *pixel* but is subsampling the image dimensions, namely sub-sampling *pixels* from each *image*, as is explained in Chaddha (column 4, lines 54-60).

Like Astle, Chaddha sub-samples YUV signals, prefilters the signals into blocks, transforms the blocks to obtain DCT coefficients. Next coefficients are converted to indexes into a TSVQ codebook, the *codebook indices*, not "*a number of pixels bits*", are run length encoded (see Chaddha column 4 line 61 through column 9 line 59 for the lengthly disclosure of this complex process.

During the transmission of the video stream, the decoder can variably select from a plurality of pre-encoded streams, but cannot change the "number of pixels bits" being encoded.

Each of the elements of Chaddha's elements is very different than those of applicant's invention.

Further Chaddha discloses a rather complicated compression method that employs a form of block-based, lossy transform steps, followed by one or quantization steps, and entropy encoding steps. This reference differs significantly from the subject invention and teaches away from the simple, fast, one-the-fly, one pass, clinically lossless method and apparatus of the present invention.

4. Walsh and Agarwal are Complex, Block-based, Lossy Compression Methods

Walsh and Agarwal both claim priority from the same provisional patent application, namely 60/001,369. For this reason the both disclosed what appear to be 28 identical figures, which each claim different aspects of the original disclosure.

Walsh and Agarwal both disclose a complex, block based, lossy compression method. Specifically, they sub-sample YUV signals, perform a wavelet transform to separate the Y component in multiple bands (Fig 4 of both). Each band is converted to motion vectors (Fig 6, 602 to 604) and the motion vectors are differenced to ouput interband differences (Fig 6, 604 to 606). A forward block transform (606) outputs coefficients. The quantizer (608) outputs quantized coefficients. Then, the *quantized coefficients*, not "*a number of pixels bits*", are run length encoded. After the *quantized coefficients* run length encoding step, a variable length coding step is used to generate an encode band. Walsh and Agarwals's complex process to produce quantized coefficients is different than Applicants "sub-sampling a number of pixel bits" from each single pixel. Walsh and Agarwal's encoding circuit (Fig 5) for counting repeated instances of *quantized coefficients* (Fig 6 is not the same as Applicant's "encoding circuit for counting repeated instances of a *[sub-sampled] pixel value* when scanning said plurality of pixels". In fact the Agarwal's counting cannot occur *while scanning pixels* on the fly because the entire frame must be digitized, before being transformed into blocks, before the blocks can be quantized, before the run length coder can start counting.

Further, Agarwal's disclosure regarding 8 bit RGB component signals (column 4, lines 10—15) is not the same as Applicant's "sub-sampling a number of pixel bits" because the all 24 bits are preserved in the plurality of component bands. (See discussion above regarding sub-sampling).

Each of the elements of Walsh and Agarwal is very different than those of applicant's invention.

Further Astle discloses a rather complicated compression method that employs a form of block-based, lossy transform steps, followed by one or more quantization steps, and entropy encoding steps. These references differ significantly from the subject invention and teach away from the simple, fast, one-the-fly, one pass, clinically lossless method and apparatus of the present invention.

5. Reliance on Federiksen

In this office action, Federiksen is relied on to teach (in combination with Astle) pixel bits being extracted from the most significant bits of each color component. Federiksen discloses a method of extracting the most significant bits of each color component, namely Y, (R-Y)av, and (B-Y) av. (Fig 3a and column 7, lines 58-62). However, two of the components, (R-Y)av and (B-Y)av, are not from a single pixel but are from "two distinct pairs of (R-Y) and (B-Y) chrominance components with at single pair of chrominance values (R-Y)av and (B-Y)av defined by the respective averages of the pair of chrominance value extracted earlier" (column 6, lines 53 through 57). Federiksen's two of three color components averaged over a pair of pixels are not the same equivalent to Applicant's color components wherein said number of pixel bits are selected from a single pixel.

Federiksen's color components are not the same or equivalent to those of applicant's color components for a single pixel.

6. Reliance on Anastassiou

In this office action, Anastassiou is relied on to teach (in combination with Astle) the step where in the image is "enlarged by stretching prior to said displaying." The office action cites Anastassiou column 2, lines 56-61 which reads *"identifying edge pixels from an encoding or decoding of the first bit plane of the image; locating the edge pixels and generating at least a single bit for each of the edge pixels; and indicating whether each edge pixel has a black/white intensity value or a gray*

intensity value by the binary state of the single bit based upon a comparison of the corresponding bits in the first and second bit planes." This sentence fragment is not the same as Applicant's claim element "wherein said images are enlarged by stretching prior to said displaying."

Further, Anastassiou is limited to compression of black and white text or line art drawings which are encoded into ***two*** single ***bit*** planes. Applicant's invention uses a ***single*** encoded data buffer for a grayscale or color plane composed of a ***plurality of bits***.

7. Reliance on Pinder

In this office action, Pinder is relied on to teach (in combination with Astle) the concept of encrypting a bitstream (column 9, lines 35-38). This reference is simply states "The modulator 36 also includes an encryptor 420 for encrypting the bit stream, a signal modulator 430 for modulating the bit stream, and an up converter 440 for producing output 445." This reference is a poor reference because it is vague and weak and should be construed narrowly. Further, Applicant's encryption is a single integrated step which occurs simultaneously with the sub-sampling step prior to the run length encoding. In Pinder the input bit stream has already been encoded.

Pinder encryptor are not the same or equivalent to those of applicant's encryption prior to forming the bit stream.

8. Astle and Federiksen Combined Do Not Teach the Invention

Astle does not teach the elements of applicant's invention but instead teaches away from them. Federiksen does not teach the elements of applicant's invention but instead teaches away from them. If Astle's elements were combined with Federiksen's element, applicant's invention would not have been taught.

9. Astle and Federiksen, Anastassiou, or Pinder Do Not Contain Any Justification To Support Their Combination, Much Less in the Manner Proposed

With regard to the proposed combination of Astle and Federiksen, Astle and Anastassiou, or Astle and Pinder, it is well known that in order for any prior art references themselves to be validly combined for use in a prior art 103 rejection, the *references themselves* (or some other prior art) must suggest they be combined (In re Semaker, 217 U.S.P.Q. 1, 5 (C.A.F.C. 1983)). "There must be some reason for the combination other than hindsight gleaned from the invention itself... Something in the prior art must suggest the desirability and thus the obviousness of

making the combination." (Uniroyal, Inc. v. Rudkin-Wiley Corp., 5 U.S.P.Q. 2d 1434 (C.A.F.C. 1988).

There are no teachings in Astle and Federiksen to combine. Astle did not suggest that it would be desirable to provide Federiksen's extraction of the most significant bits of each color component. Federiksen did not suggest the use of Astle's block determination. Astle was filed long after Federiksen was published and failed to teach the use of Federiksen's method. Astle does not teach that it would be desirable to use any of the details of Federiksen's most significant bit extraction method.

There are no teachings in Astle and Anastassiou to combine. Astle did not suggest that it would be desirable to provide Anastassiou's methods Anastassiou did not suggest the use of Astle's block determination. Astle was filed long after Anastassiou was published and failed to teach the use of Anastassiou's method. Astle does not teach that it would be desirable to use any of the details of Anastassiou's method..

There are no teachings in Astle and Pinder to combine. Astle did not suggest that it would be desirable to provide Pinder's encryptor. Pinder did not suggest the use of Astle's block determination. Pinder was filed long after Astle was published and failed to teach the use of Astle's methods. Pinder does not teach that it would be desirable to use any of the details of Astle's block determination method.

10. Each of the Combined References Are Individually Complete

Each reference is complete and functional in itself, so there would be no reason to use parts from or add or substitute parts of any reference.

11. Each Combined Reference Take Different Approaches and Teach Away for Each Other

The references take mutually exclusive paths and reach different solutions to a similar problem. Since they teach away from each other, it would not be logical to combine them.

12. Combining Astle and Federiksen, Anastassious, or Pinder, Respectively, is Improper

Applicant therefore submits that combining Astle with any of the other three references (Federiksen, Anastassiou, or Pinder) is not legally justified and is therefore improper. Thus applicant submits that the rejection on these references is also improper and should be withdrawn.

13. Summary

In summary, the cited references do not teach all of the elements and limitations of claims 1-21 as amended, and thus, do not anticipate nor make obvious claims 1-21 of the present invention. Neither Astle, Chaddha, Walsh, or Agarwal, individually teach all the elements and limitations of claims 1-21. It is improper to combine the teachings in the office actions proposed combinations regarding claims 4, 10, 13, 20 and 21. Applicant's invention produces new and unexpected results.

C. Applicant's Claims Are Patentable Over the Cited References for Anticipation

Astle

Claims 1 (including Chaddha)

Regarding Astle, Applicant respectfully disagrees with the office action's conclusion that Astle teaches "(a) sub-sampling a *number of pixel bits*." As stated above Astle Fig. 4, 404 shows YUV 4:1:1 subsampling (Fig 4, 404) across a plurality of pixels is different than Applicants "sub-sampling a number of pixel bits" from each single pixel. Astle lacks the essential function of selecting a code based on a number of bits from each pixel. Astle teaches instead a block-based, transformed method of coding *coefficients*.

Applicant respectfully disagrees with the office action's conclusion that Astle teaches "(b) run-length encoding repeated instances of the *number of pixel bits*." Astle Fig. 4, 412, is run length encoded inter/intra *encoded signals* that have already been encoded with the above described block-based, transformed method of coding *coefficients*. Astle lacks the essential element "run length encoding repeated instances of said *number of pixel bits*". Astle teaches instead run length encoding prefiltered, discrete cosine transformed, quantized, *block data coefficients* as its input the run-length encoder 412 as shown is Fig 5 (Fig 4's 412 is the same as Fig 5's 412).

Finally, Astle fails to clearly disclose an *encoded data buffer*. Astle's "host processor and memory" (Fig 2, 202) is not the same as Applicant's "encoded data buffer".

Regarding Chaddha, Applicant respectfully disagrees with the office action's conclusion that Chaddha teaches "(a) sub-sampling a *number of pixel bits*." As stated above Chaddha's sub-

sampling (column 4, lines 54-60) is more appropriately characterized as sub-sampling *pixels* from each image rather than "sub-sampling a number of *pixel bits*" from each *pixel*. Chaddha lacks the essential function of sub-sampling a number of bits from each pixel. Chaddha teaches away from Applicant's invention by requiring the decompression method (Fig 2, 270) to be present on the encoder.

For all these reasons, Applicant submits that claim 1 is patentable over the references. Note that the amendment to claim 1 is to correct an informality not to overcome any prior art.

Claim 7

As stated above with respect to claim 1, Astle's run-length encoded codes do not directly contain pixel values and cannot be decoded into "*values and run-lengths*". Instead Fig 14 shows that the output of the run-length decoder 1306 are *DCT indices* that must be inverse quantized (1312) to obtain DCT coefficients which are then summed to obtain frame coefficients (1316) which are then inverse DCT transformed (1320) to obtain frame pixels. Astle's frame pixels have been distorted by the loss of data in the transform and quantizing steps and must first be post filtered (1322) before they can be displayed. Astle lacks the necessary claim element of decoding "values and run-lengths". Astle's elements are not the same or equivalent to Applicant's elements. Further, Astle teaches away from the teaches away from the simple, fast, one-the-fly, one pass, clinically lossless method and apparatus of the present invention, and therefore does not make Applicants invention obvious.

For all these reasons, Applicant submits that claim 7 as amended is patentable over the reference.

Claim 2 and 9

Claim 2 adds additional limitations on claim 1. Claim 9 adds additional limitations on claim 7. Applicant submits that claims 2 and 9 should now be allowable because they are dependent on claim 1 and claim 7, respectively, and each independent claim should now be allowable for the reasons cited above.

Further, as discussed above regarding sampling and sub-sampling, although image dimensions of 320 x 240 were well known in the art, claims 2 and 9 should both be interpreted in

light of the specification which states "This 640 x 480 pixel array is a still graphical image that is considered to be full frame" and understand that this claim means that the pixels are dimensionally sub-sampled from an image where the original sample is larger than 320 x 240. This is consistent with the description and example of sub-sampling provided in the specification on page 7, which reads:

> Sub-sampling is the selection of a subset of data from a larger set of data. For example, when every other pixel of every other row of a video image is selected, the resulting image has half the width and half the height.

For all these reasons, Applicant submits that claims 2 and 9 as written are patentable over the reference.

Claim 3

Claim 3 adds additional limitations on claim 1. Applicant submits that claim 3 should now be allowable because it is dependent on claim 1 which should now be allowable for the reasons cited above.

Further, as discussed above regarding Applicant's terminology regarding. Astle's YUV component signal is represented by 24 bits per pixel. However these 24 bits are the size of the original sample. Astle fails to teach subsampling from a larger sample to obtain 24 bits. Astle lacks the essential claim elements of "sub-sampling a number of pixel bits... wherein said number of pixel bits is ... 24". Claim 3 should be interpreted in light of the specification, including the disclosure of the provisional patent application, which teaches that the number of sub-sampled pixel bits N is smaller than the original number of sample bits W. For example, an image could be scanned at 36 bit resolution (12 bits for each red, green, and blue color component, or a full 36 bits of grayscale), and the 36 bit sample could be sub-sampled to 24 bits (8 bits from each color component or 24 bits of grayscale). Astle's 24 bit YUV sample lacks the essential claim elements of the present invention.

For all these reasons, Applicant submits that claim 3 as written is patentable over the reference.

Claim 8

Claim 8 adds additional limitations on claim 7. Applicant submits that claim 8 should now be allowable because it is dependent on claim 7 which should now be allowable for the reasons cited above.

Further, as discussed above regarding claim 7, Astle's series of images are not the same as Applicant's series of images because Astle's decoded, inverse quantized, inverse discrete consine transformed, and post filtered images have been distorted by the lossy compression method. Applicants images are clinically lossless in that they do contain the arbitrary loss and image distortions and other compression artifacts introduced by the transform and quantizing steps and blurred by the post filtering step.

For all these reasons, Applicant submits that claim 8 as written is patentable over the reference.

Claim 19

Claim 19 adds additional limitations on claim 1. Applicant submits that claim 19 should now be allowable because it is dependent on claim 1 which should now be allowable for the reasons cited above.

Further, the office action misunderstands the specification and the claim. Claim 19 is not suggesting that anything as complex as MPEG standard be used with applicants inventions. Claim 19 should be interpreted in light of the specification that discusses a number of entropy encoded techniques (such as Huffman coding or arithmetic, finite context, or adaptive coding) that could be used as an additional step to further compress the "said [run length encoded data] buffer". Astle teaches use of a variable length entropy encoder (Fig 4 and 5, 414) but as stated above in reference to claim 1, Applicant's input to the entropy encoding step is not the same or equivalent to Astle's run-length encode block data (see Fig 5).

For all these reasons, Applicant submits that claim 19 as written is patentable over the reference.

Chaddha

Claim 1

Regarding claim 1, Chaddha has been addressed above along with Astle's claim 1 rejection.

Claims 17 and 18

Claim 17 adds additional limitations on claim 1. Claim 18 adds additional limitations on claim 1. Applicant submits that claims 17 and 18 should now be allowable because they are dependent on claim 1 which should now be allowable for the reasons cited above.

Regarding Chaddha, Applicant respectfully disagrees with the office action's conclusion that Chaddha teaches that "the *number of pixel bits (relates to size of a display area such as 160 x120, 320 x 240)* are *variably altered* by a receiver (decoder, 40) of the encoded data" As stated above, Chaddha's sub-sampling (Fig 3, column 4, lines 54-60) is more appropriately characterized as sub-sampling *pixels* from each image rather than "sub-sampling a number of *pixel bits*" from each *pixel*. The office action misunderstands the reference. Chaddha image dimensions relates to the *number of pixels* not the *number of pixel bits*. With regard *settings* or *number of pixel bits* being *variably altered by the receiver*, Chaddha does not teach that the claimed encoder's settings are variably altered during the real time compression method as claimed by Applicant. Rather Chaddha teaches away from real time compression where every possible setting is *fixedly* preprocessed. In Chaddha, during transmission, the decoder receives from one or more preset, stored video streams, but does not remotely control the encoding process. Chaddha lacks the essential element of remote control from a receiver. Chaddha also teaches away from Applicant's invention by requiring the decompression method (Fig 2, 270) to be present on the encoder.

For all these reasons, Applicant submits that claims 17 and 18 as written are patentable over the reference.

Walsh

Claims 5 and 6

Claim 5 has been rewritten as depended claim to overcome the informality objection. This amendment was not made to overcome the prior art. Due to the amendment office action's rejection is mute, and applicant respectfully requests reconsideration of the amended claim.

Claim 5 is now dependent on claim 1. Claim 6 is dependant on claim 5 and indirectly dependant on claim 1. Applicant submits that claims 5 and 6 should now be allowable because they are dependent on claim 1 which should now be allowable for the reasons cited above.

For all these reasons, Applicant submits that claims 5 and 6 as amended are patentable over the reference.

Agarwal

Claims 11, 14, and 15

The claim term "pixel value" should be interpreted in light of the specification. The summary of the invention discloses "run-length encoding the sub-sampled pixel values". Thus the claim language "an encoding circuit for counting repeated instances of a pixel value" should be understood to mean a sub-sampled or filtered pixel value which is analogous to the "sub-sampling a number of pixel bits" language of claim 1. Applicant's "*pixel value*" is not the same as Agarwal's (or Walsh's) "*intra frame encoding*" which, as discussed above. Both Agarwal and Walsh sub-sample YUV signals, perform a wavelet transform to separate the Y component in multiple bands (Fig 4 of both). Each band is converted to motion vectors (Fig 6, 602 to 604) and the motion vectors are differenced to ouput interband differences (Fig 6, 604 to 606). A forward block transform (606) outputs coefficients. The quantizer (608) outputs quantized coefficients. Then, the *quantized coefficients*, not "*pixel values*", are run length encoded.

Agarwal lacks the necessary claim 11 element of "counting repeated instances of a *pixel value*". For all these reasons, Applicant submits that claim 11 as amended is patentable over the reference. Note that the amendment of claim 11 is not to overcome prior art, but is to correct an informality.

Claim 14 is dependent on claim 11. Claim 15 is dependant on claim. Applicant submits that claims 5 and 6 should be allowable because they are dependent on claim 11 which should now be allowable for the reasons cited above.

For all these reasons, Applicant submits that claims 11, 14, and 15 as amended/written are patentable over the reference.

Claim 12

Claim 12 adds additional limitations on claim 11. Applicant submits that claim 12 should now be allowable because it is dependent on claim 11 which should now be allowable for the reasons cited above.

Further Agarwal's disclosure regarding 8 bit RGB component signals (column 4, lines 10—15) is not the same as Applicant's sub-sampled pixel value (as discussed above in regards to claim 11) because the all 24 bits are preserved in the plurality of component bands. (See discussion above regarding sub-sampling and if reference to claim 3).

Agarwal lacks the necessary claim 11 elements of "counting repeated instances of a *pixel value*" and of *variably selecting* "*the number of pixel bits*". For all these reasons, Applicant submits that claim 12 as written is patentable over the reference.

Claim 16

As discussed above in relation to claim 11, the claim term "pixel value" should be interpreted in light of the specification. The summary of the invention discloses "run-length encoding the sub-sampled pixel values". Thus the claim language "a decoding circuit which can decode the encoded data and output a stream of pixel values" should be understood to mean a sub-sampled or filtered pixel value which is analogous to the "sub-sampling a number of pixel bits" language of claim 1. Applicant's "*encoded data*" and "*pixel values*" is not the same as, or equivalent to, Agarwal's (or Walsh's) "*quantized coefficients*" (Fig 18 between 1804 and 1806) and "*run values*", which are zig zag run-length encoded Huffman codes (Fig 18, 1802 through 1804), respectively.

Agarwal lacks the necessary claim 16 elements of "*encoded data*" and "*pixel values*". For all these reasons, Applicant submits that claim 16 as written is patentable over the reference.

D. Applicant's Claims Are Patentable Over the Cited References for Obviousness

Astle in View of Frederiksen

Claim 4, 10, and 13

Claim 4 as written adds additional limitations on claim 3. Claim 10 as written adds additional limitations on claim 7. Claim 13 as written adds additional limitations on claim 12. Applicant submits that claims 4, 10, and 13 should now be allowable because each is dependent on claims 3, 7, and 12, respectively, which each should be allowable for the reasons cited above.

Further, Applicant respectfully disagrees with the office action's conclusion that it would have been obvious to employ Astles compression method with Federiksen's most significant

pixel extraction to create the subject invention. As explained above, neither applicant's compression method or its essential elements are taught by either Astle or Frederick, either individually or in combination. It would also be improper to combine the references.

For all these reasons, Applicant submits that claims 4, 10, and 13 as written are patentable over the references.

Astle in View of Anastassiou

Claim 20

Claim 20 as written adds additional limitations on claim 8. Applicant submits that claim 20 should now be allowable because each is dependent on claim 8 which each should be allowable for the reasons cited above.

Further, Applicant respectfully disagrees with the office action's conclusion that it would have been obvious to employ Astles compression method with the Anastassiou reference. As explained above, neither applicant's compression method or its essential elements are taught by either Astle or Anastassiou, either individually or in combination. It would also be improper to combine the references.

For all these reasons, Applicant submits that claim 20 as written is patentable over the references.

Astle in View of Pinder

Claim 21

Claim 21 as written adds additional limitations on claim 1. Applicant submits that claim 20 should now be allowable because each is dependent on claim 1 which each should be allowable for the reasons cited above.

Further, Applicant respectfully disagrees with the office action's conclusion that it would have been obvious to employ Astles compression method with Pinder encrypting of an already encoded bitstream (as explained above in reference to Pinder). As explained above, neither applicant's compression method or its essential elements are taught by either Astle or Pinder, either individually or in combination. It would also be improper to combine the references.

For all these reasons, Applicant submits that claim 21 as written is patentable over the references.

E. Amended Claims

Please amend claim 1 according to the correction requested in the office action.

Please amend claim 5 as written in dependent form to provide a proper antecedent for the claim element "said encoded data buffers". Note that this change was not made to overcome prior art references. Prior art rejections of the elements of claim 5 are transversed above, without amending any element.

Please amend claim 7 by replacing the trailing "; and" with a period. This is an obvious clerical error.

Please amend claim 11 by replacing the word "a" with the word "an" at the beginning of 11(c). This is an obvious clerical error.

F. New Claims

Applicant submits claims 22 through 25 that should be allowable as written

Claims 22 through 25 have been added. The features of these claims are novel because, as stated various times above, they are not seen, as specifically claimed in the present invention, in the prior art. Specifically, the present invention embodies systems and methods that allow for fast, real time, and high quality compression of video images. The features of these claims are not obvious and produce superior and or unexpected results from what is found in the prior art. Specifically, by improving the effective compression without the introduction of distortion by conventional compression methods, an number of benefits result, including but not limited to being able to receive an image from a video stream over a connection medium in real-time that could not be received without substantially more distortion, space, bandwidth, or cost.

G. Other Amendments to the Specification

In addition to a number of self-explanatory clerical errors that are corrected, the specification is amended to provide the proper antecedent basis for the claimed subject matter.

No new matter has been added. Support for these changes is found in the original specification and claims, including the provisional application and referenced related co-pending applications.

H. Description of New Figures

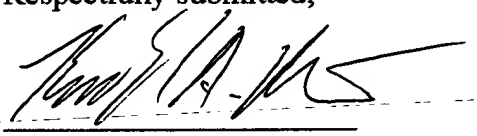
Applicant has added terms for each of the new reference numbers. Applicant respectfully requests instructions what a level of description, if any, may be added to the specification for the new figures that provide the require antecedent the claims.

RECONSIDERATION REQUESTED

Attached hereto is a marked-up version of the changes made to the specification by the current amendment. The attached page is captioned **"Version with Markings to Show Changes Made"**. Also, a clean copy of the specification and claims with the new claims is attached. Drawing sheets 14 through 23 with Figures 12A through 18 are submitted as well.

The undersigned respectfully submits that, in view of the foregoing amendments and remarks, the rejections of the claims raised in the Office Action dated September 26, 2002 have been fully addressed and overcome, and the present application is believed to be in condition for allowance. It is respectfully requested that this application be reconsidered, that these claims be allowed, and a Notice of Allowance is respectfully requested. If it is believed that a telephone conversation would expedite the prosecution of the present application, or clarify matters with regard to its allowance, the Examiner is invited to call the undersigned at 408-739-9517.

Respectfully submitted,



Pearl Harbor, HI

Kendyl A. Roman
Phone: 408-739-9517

Date: March 26, 2003

VERSION WITH MARKING SHOWING CHANGES MADE

Patent Application of

Kendyl A. Román

for

TITLE: VARIABLE GENERAL PURPOSE COMPRESSION FOR VIDEO IMAGES (ZLN)

CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims priority under 35 U.S.C. § 119(e) of the co-pending U.S. provisional application Serial Number 60/113,051 filed on 1998 December 21, and entitled "METHODS OF ZERO LOSS (ZL) COMPRESSION AND ENCODING OF GRAYSCALE IMAGES." The provisional application Serial Number 60/113,051 filed on 1998 December 21 and entitled "METHODS OF ZERO LOSS (ZL) COMPRESSION AND ENCODING OF GRAYSCALE IMAGES" is also hereby incorporated by reference.

A U.S. patent application ~~to be~~Serial Number 09/470,566, filed on 1999 December 22, and entitled GENERAL PURPOSE COMPRESSION FOR VIDEO IMAGES (RHN)", to be known as the "RHN" ~~algorithm~~method, has claims that combine some of the elements of the present invention in a different combination. The RHN application claims a priority date based on a co-pending U.S. provisional application Serial Number 60/113,276 filed on 1998 December 23. The provisional application Serial Number 60/113,276 filed on 1998 December 23 is also hereby incorporated by reference. The application- Serial Number 09/470,566, filed on 1999 December 22, and entitled GENERAL PURPOSE COMPRESSION FOR VIDEO IMAGES (RHN)" as amended is also hereby incorporated by reference.

My co-pending U.S. patent application Serial Number 90/312,922 filed on 1999 May 17, and entitled "SYSTEM FOR TRANSMITTING VIDEO IMAGES OVER A

COMPUTER NETWORK TO A REMOTE RECEIVER" describes an embodiment of the invention of the RHN method, as well as a system for practicing the compression method. U.S. patent application Serial Number 90/312,922, filed on 1999 May 17, and entitled "SYSTEM FOR TRANSMITTING VIDEO IMAGES OVER A COMPUTER NETWORK TO A REMOTE RECEIVER" is also hereby incorporated by reference.

My co-pending U.S. patent application, serial number _____, filed on 1999 December 20, and entitled "ADDING DOPPLER ENHANCEMENT TO GRAYSCALE COMPRESSION (ZLD)" describes an invention that is related to this application. U.S. patent application Serial Number _____, filed on 1999 December 20, and entitled "ADDING DOPPLER ENHANCEMENT TO GRAYSCALE COMPRESSION (ZLD)" as amended is also hereby incorporated by reference.

BACKGROUND – FIELD OF THE INVENTION

This invention relates to data compression, specifically to the compression and decompression of video images.

BACKGROUND–DESCRIPTION OF PRIOR ART

In the last few years, there have been tremendous advances in the speed of computer processors and in the availability of bandwidth of worldwide computer networks such as the Internet. These advances have led to a point where businesses and households now commonly have both the computing power and network connectivity necessary to have point-to-point digital communications of audio, rich graphical images, and video. However the transmission of video signals with the full resolution and quality of television is still out of reach. In order to achieve an acceptable level of video quality, the video signal must be compressed significantly without losing either spatial or temporal quality.

A number of different approaches have been taken but each has resulted in less than acceptable results. These approaches and their disadvantages are disclosed by Mark Nelson in a book entitled The Data Compression Book, Second Edition, published by M&T Book in 1996. Mark Morrision also discusses the state of the art in a book entitled ~~The Magic of Image Processing~~ The Magic of Image Processing, published by Sams Publishing in 1993.

Video Signals

Standard video signals are analog in nature. In the United States, television signals contain 525 scan lines of which 480 lines are visible on most televisions. The video signal represents a continuous stream of still images, also known as frames, that are fully scanned, transmitted and displayed at a rate of 30 frames per second. This frame rate is considered full motion.

–A television screen has a 4:3 aspect ratio.

When an analog video signal is digitized each of the 480 lines are sampled 640 times, and each sample is represented by a number. Each sample point is called a picture element, or pixel. A two dimensional array is created that is 640 pixels wide and 480 pixels high. This 640 x 480 pixel array is a still graphical image that is considered to be full frame. The human eye can perceive 16.7 thousand colors. A pixel value comprised of 24 bits can represent each perceivable color. A graphical image made up of 24-bit pixels is considered

to be full color. A single, second-long, full frame, full color video requires over 220 millions bits of data.

The transmission of 640 x 480 pixels x 24 bits per pixel times 30 frames requires the transmission of 221,184,000 millions bits per second. A T1 Internet connection can transfer up to 1.54 millions bits per second. A high speed (56Kb) modem can transfer data at a maximum rate of 56 thousand bits per second. The transfer of full motion, full frame, full color digital video over a T1 Internet connection, or 56Kb modem, will require an effective data compression of over 144:1, or 3949:1, respectively.

A video signal typically will contain some signal noise. In the case where the image is generated based on sampled data, such as an ultrasound machine, there is often noise and artificial spikes in the signal. A video signal recorded on magnetic tape may have fluctuations due the irregularities in the recording media. Florescent or improper lighting may cause a solid background to flicker or appear grainy. Such noise exists in the real world but may reduce the quality of the perceived image and lower the compression ratio that could be achieved by conventional methods.

Basic Run-length Encoding

An early technique for data compression is run-length encoding where a repeated series of items are replaced with one sample item and a count for the number of times the sample repeats. Prior art shows run-length encoding of both individual bits and bytes. These simple approaches by themselves have failed to achieve the necessary compression ratios.

Variable Length Encoding

In the late 1940s, Claude Shannon at Bell Labs and R.M. Fano at MIT pioneered the field of data compression. Their work resulted in a technique of using variable length codes where codes with low probabilities have more bits, and codes with higher probabilities have fewer bits. This approach requires multiple passes through the data to determine code probability and then to encode the data. This approach also has failed to achieve the necessary compression ratios.

D. A. Huffman disclosed a more efficient approach of variable length encoding known as Huffman coding in a paper entitled "A Method for Construction of Minimum

Redundancy Codes," published in 1952. This approach also has failed to achieve the necessary compression ratios.

Arithmetic, Finite Context, and Adaptive Coding

In the 1980s, arithmetic, finite coding, and adaptive coding have provided a slight improvement over the earlier methods. These approaches require extensive computer processing and have failed to achieve the necessary compression ratios.

Dictionary-Based Compression

Dictionary-based compression uses a completely different method to compress data. Variable length strings of symbols are encoded as single tokens. The tokens form an index to a dictionary. In 1977, Abraham Lempel and Jacob Ziv published a paper entitled, "A Universal Algorithm for Sequential Data Compression" in IEEE Transactions on Information Theory, which disclosed a compression technique commonly known as LZ77. The same authors published a 1978 sequel entitled, "Compression of Individual Sequences via Variable-Rate Coding," which disclosed a compression technique commonly known as LZ78 (see U.S. Patent 4,464,650). Terry Welch published an article entitled, "A Technique for High-Performance Data Compression," in the June 1984 issue of IEEE Computer, which disclosed an algorithm commonly known as LZW, which is the basis for the GIF algorithm (see US Patents 4,558,302, 4,814,746, and 4,876,541). In 1989, Stack Electronics implemented a LZ77 based method called QIC-122 (see U.S. Patent 5,532,694, U.S. Patent 5,506,580, and U.S. Patent 5,463,390).

These lossless (method where no data is lost) compression methods can achieve up to 10:1 compression ratios on graphic images typical of a video image. While these dictionary-based algorithms are popular, these approaches require extensive computer processing and have failed to achieve the necessary compression ratios.

JPEG and MPEG

Graphical images have an advantage over conventional computer data files: they can be slightly modified during the compression/decompression cycle without affecting the perceived quality on the part of the viewer. By allowing some loss of data, compression ratios of 25:1 have been achieved without major degradation of the perceived image. The Joint Photographic Experts Group (JPEG) has developed a standard for graphical image compression. The JPEG lossy (method where some data is lost) compression algorithm first

divides the color image into three color planes and divides each plane into 8 by 8 blocks, and then the algorithm operates in three successive stages:

(a) A mathematical transformation known as Discrete Cosine Transform (DCT) takes a set of points from the spatial domain and transforms them into an identical representation in the frequency domain.

(b) A lossy quantization is performed using a quantization matrix to reduce the precision of the coefficients.

(c) The zero values are encoded in a zig-zag sequence (see Nelson, pp. 341-342).

JPEG can be scaled to perform higher compression ratio by allowing more loss in the quantization stage of the compression. However this loss results in certain blocks of the image being compressed such that areas of the image have a blocky appearance and the edges of the 8 by 8 blocks become apparent because they no longer match the colors of their adjacent blocks. Another disadvantage of JPEG is smearing. The true edges in an image get blurred due to the lossy compression method.

The Moving Pictures Expert Group (MPEG) uses a combination of JPEG based techniques combined with forward and reverse temporal differencing. MPEG compares adjacent frames and for those blocks that are identical to those in a previous or subsequent frame and only a description of the previous or subsequent identical block is encoded. MPEG suffers from the same blocking and smearing problems as JPEG.

These approaches require extensive computer processing and have failed to achieve the necessary compression ratios without unacceptable loss of image quality and artificially induced distortion.

QuickTime: CinePak, Sorensen, H.263

Apple Computer, Inc. released a component architecture for digital video compression and decompression, named QuickTime. Any number of methods can be encoded into a QuickTime compressor/decompressor (codec). Some popular codec are CinePak, Sorensen, and H.263. CinePak and Sorensen both require extensive computer processing to prepare a digital video sequence for playback in real time; neither can be used for live compression. H.263 compresses in real time but does so by sacrificing image quality resulting in severe blocking and smearing.

Fractal and Wavelet Compression

Extremely high compression ratios are achievable with fractal and wavelet compression algorithms. These approaches require extensive computer processing and generally cannot be completed in real time.

Sub-sampling

Sub-sampling is the selection of a subset of data from a larger set of data. For example, when every other pixel of every other row of a video image is selected, the resulting image has half the width and half the height.

Image Stretching

If an image is to be enlarged but maintain the same number of pixels per inch, data must be filled in the new pixels that are added. Various methods of stretching an imaging and filling in the new pixels to maintain image consistency. Some methods known in the art are dithering (using adjacent colors that appear to be blended color), -and error diffusion, "nearest neighbor", bilinear and bicubic.

SUMMARY OF THE INVENTION

In accordance with the present invention a method of compression of a video stream comprises steps of sub-sampling a video frame, and run-length encoding the sub-sampled pixel values, whereby the method can be executed in real time and the compressed representation of pixels saves substantial space on a storage medium and require substantially less time and bandwidth to be transported over a communications link. The present invention includes a corresponding method for decompressing the encoded data.

Objects and Advantages

Accordingly, beside the objects and advantages of the method described in our patent above, some additional objects and advantages of the present invention are:

(a) to provide a method of compressing and decompressing video signals so that the video information can be transported across a digital communications channel in real time.

(b) to provide a method of compressing and decompressing video signals such that compression can be accomplished with software on commercially available computers without the need for additional hardware for either compression or decompression.

(c) to provide a high quality video image without the blocking and smearing defects associated with prior art lossy methods.

(d) to provide a high quality video image that suitable for use in medical applications.

(e) to enhance images by filtering noise or recording artifacts.

(f) to provide a method of compression of video signals such that the compressed representation of the video signals is substantially reduced in size for storage on a storage medium.

(g) to provide a level of encryption so that images are not directly viewable from the data as contained in the transmission.

DRAWING FIGURES

In the drawings, closely related figures have the same number but different alphabetic suffixes.

Fig 1 shows the high level steps of compression and decompression of an image.

Fig 2A to 2H show alternatives for selecting a pixel value for encoding.

Fig 3A shows the variable encoding format.

Fig 3B shows an example of a code where N is 5 bits wide and U is 3 bits wide.

Fig 4A shows the flowchart for the compression method.

Fig 4B shows an image and a corresponding stream of pixels.

Fig 5A to 5C shows the formats for the run-length encoding of the RHN ~~algorithm~~method.

Fig 6 shows a series of codes and the resulting encoded stream.

Fig 7 shows a series of codes and the resulting encoded stream of the RHN ~~algorithm~~method.

Fig 8A shows examples of variable formats.

Fig 8B shows a format that preserves 9 bits of color.

Fig 9 shows the flow chart for the decompression method.

Figs 10 shows image stretching by interpolation.

Figs 11A and 11B show an encryption table and a decryption table.

Figs 12A and 12B show an machines for compressing and decompressing, respectively.

Fig 12C shows a compressor and decompressor connected to a storage medium

Fig 12D shows a compressor and decompressor connected to a communications channel.

Fig 13A shows elements of a compressor.

Fig 13B shows an embodiment of an encoding circuit.

Fig 13C shows a generic pixel sub-sampler.

Figs 13D through 13J show embodiments of pixel sub-samplers.

Figs 14A through 14c shows embodiments of a machine element for variably altering the number of bits.

Fig 15 shows elements of a decompressor.

Fig 16A shows elements for setting width, height, frame rate, brightness, and contrast which are variably altered by a receiver.

Fig 16B shows elements for setting the number of pixel bits which are variably altered by a receiver.

Fig 17 shows a lossless compression step for further compression an encoded data buffer.

Fig 18 shows images being enlarged by stretching.

Reference Numerals in Drawings

100	compression steps	110	sub-sampling step
130	encoding step		
140	encoded data	150	decompression steps
160	decoding step	180	image reconstitution step
200	32 bit pixel value		
202	blue channel	204	green channel
206	red channel	208	alpha channel
210	24 bit pixel value	212	blue component
214	green component	216	red component
220	RGB averaging diagram	222	blue value
224	green value	226	red value
228	averaged value	230	blue selection diagram

232	blue instance	234	green instance
236	red instance	240	selected blue value
250	green selection diagram	260	selected green value
270	red selection diagram	280	selected red value
290	grayscale pixel	292	grayscale blue
294	grayscale green	296	grayscale red
298	selected grayscale value	299	filtered pixel value
300	N	301	U
302	W	310	pixel bit 7
312	pixel bit 6	314	pixel bit 5
316	pixel bit 4	318	pixel bit 3
320	pixel bit 2	322	pixel bit 1
324	pixel bit 0	325	8 bit pixel
330	5 bit sample		
332	sample bit 4	334	sample bit 3
336	sample bit 2	338	sample bit 1
340	sample bit 0	350	3 low order bits
360	formatted code	380	3 bit count value
400	encode flowchart		
402	encode entry	403	encode initialization step
404	get pixel step	405	get value step
406	lookup encoded value step	408	compare previous
410	increment counter step	412	check count overflow
414	new code step	416	check end of data
418	set done	420	counter overflow step
422	check done	428	encode exit
430	image	440	image width
450	image height	460	pixel stream
500	code byte	510	flag bit
520	repeat code	530	count
550	data code	560	wasted bits
565	data bit 6		

570 data bit 5
575 data bit 4
580 data bit 3
585 data bit 2
590 data bit 1
595 data bit 0
610 decimal values
620 first value
622 second value
624 third value
626 fourth value
628 fifth value
630 sixth value
632 seventh value
640 binary code
650 first byte
651 first data
652 first count
653 second byte
654 second data
655 second count
656 third byte
657 third data
658 third count
740 RHN binary code
803 ZL3 format
804 ZL4 format
805 ZL5 format
808 ZL8 format
809 ZL9 format
812 ZL12 format
820 ZL9C format
900 decode entry
901 decode initialize step
902 get code step
908 decode lookup step
909 check zero count
910 place pixel step
914 reset counter step
916 check length
918 decode exit
1000 encryption key
1010 first adjacent pixel
1012 second adjacent pixel
1010 first subsequent adjacent pixel
1012 second subsequent adjacent pixel
1052, 1054, 1056, 1058, 1060 interpolated pixels
1100 encryption table
1110 decryption table

1200 video frames
1205a first video frame
1205b second video frame
1205n nth video frame
1210 compressor
1215 video signal
1220 series of encoded data
1225 encoded data buffer
1225a first encoded data
1225b second encoded data

1225n nth encoded data
1230a first received encoded data
1230b second received encoded data
1230n nth received encoded data
1238 received encoded data
1235 encoded data stream
1240 I/O device
1245 input encoded data stream
1250 decompressor
1260a first decoded video frame
1260b second decoded video frame
1260n nth decoded video frame
1268 decoded video frames
1270 video sequence
1280 storage medium
1290 communications channel
1310 video digitizer
1320 path 1320
1330 video memory
1331 scan
1332 pixel index
1340 path 1340
1350 encoding circuit
1360 path 1360
1370 encoded data
1380 pixel sub-sampler
1380a 24 to 5 bit sub-sampler
1380b 24-bit RGB to 5 bit sub-sampler
1380c 32-bit RGB to 5 bit sub-sampler
1380d color 9-bit sub-sampler
1380e YUV sub-sampler
1380f 36-bit RGB to 24-bit sub-sampler
1380g 15-bit sub-sampler
1380 pixel sub-sampler
1380 pixel sub-sampler
1380 pixel sub-sampler
1382 pixel extractor
1383 value path
1384 coder
1385 path 1385
1390 data/count
1392 code index
1395 path 1395
1400 24-bit to variable bit sub-sampler
1401 generic 3-bit sub-sampler
1402 generic 4-bit sub-sampler
1403 generic 8-bit sub-sampler
1404 generic 10-bit sub-sampler

1410 number of bits selector
1420 number of bits indicator
1430 36-bit to variable bit sub-sampler
1440 24/36 bit variable bit sub-sampler
1450 second selector
1460 selection logic
1510 decoding circuit
1520 decoded pixel values
1530 decoder pixel index
1540 image memory
1600 transmitter
1610 receiver
1615 setting control path
1620 frame sub-sampler
1621 path 1621
1630 selected frame
1632 pixel from frame
1640 transmitter pixel sub-sampler
1642 path 1642
1650 run length encoder
1660 settings
1661 brightness
1662 contract
1663 height
1664 width
1665 frame rate
1670 frame selector
1675 frame select indicator
1680 number of pixel bits setting
1700 run-length encoding step
1710 run-length encoded output
1720 further lossless compression step
1730 further lossless compression output
1800 unstretched frame
1810 enlarged image
1820 stretching step

DESCRIPTION OF THE INVENTION

Fig 1–Compression and Decompression Steps

Fig 1 illustrates a sequence of compression steps 100 and a sequence of decompression steps 150 of the present invention. The compression steps 100 comprise a sub-sampling step 110 and an encoding step 130. After completion of the compression steps 100, a stream of encoded data 140 is output to either a storage medium or a transmission

channel. The decompression steps 150 comprise a decoding step 160 wherein the stream of encoded data 140 is processed and an image reconstitution step 180.

Figs 2A to 2H Selecting Pixel Values for Encoding

Figs 2A to 2G illustrate alternatives for selecting a pixel value for encoding. The sub-sampling step 110 (Fig 1) includes sub-sampling of a pixel value to obtain a variable selected number of bits.

Video digitizing hardware typical has the options of storing the pixel values as a 32 bit pixel value 200 or a 24 bit pixel value 210, shown in Fig 2A and Fig 2B, respectively. The 32 bit pixel value 200 is composed of a blue channel 202, a green channel 204, a red channel 206, and an alpha channel 208. Each channel contains of 8 bits and can represent 256 saturation levels for the particular color channel. For each channel the saturation intensity value of zero represents the fully off state, and the saturation intensity value of "255" represents the fully on state. A common alternative not shown is a sixteen bit format where the three color channels contain 5 bits each and the alpha channel is a single bit. The present invention anticipates the use of the color channels of 16 bit pixel value is a manner substantially the same as the 32-bit pixel value 200 except the number of bits per channel is 5 instead of 8.

The 24 bit pixel value 210 is composed of a blue component 212, a green component 214, and a red component 216. There is no component for the alpha channel in the 24 bit pixel value 210. Regardless of the structure, the blue channel 202 is equivalent to the blue component 212, the green channel 204 is equivalent to the green component 214, and the red channel 206 is equivalent to the red component 216.

In the present invention, the 32 bit pixel value 200 alternative is preferred due to the consistent alignment of 32 bit values in most computer memories; however for simplicity of illustration the alpha channel 208 will be omitted in Fig 2C to 2G.

If the video signal is digitized in color, the three color components may have different values. For example in Fig 2C, a RGB averaging diagram 220 illustrates a blue value 222 of 35 decimal, a green value 224 of 15, and a red value 226 of 10. One alternative is to sub sample from 24 bits to 8 bits by averaging the three color values to obtain an averaged value 228 that, in this example, has the value of 20. (10+15+35)/3 = 20. This will

produce a grayscale image. Alternatively, a color image can be preserved by sampling bits from each color component (see Fig 8B).

Fig 2D illustrates another alternative for selecting an 8 bit value in a blue selection diagram 230. In this example, a blue instance 232 has the value of 35, a green instance 234 has the value of 15, and a red instance 236 has the value of 10. In this alternative the blue instance 232 is always selected as a selected blue value 240.

Fig 2E illustrates another alternative for selecting an 8 bit value in a green selection diagram 250. In this alternative the green instance 234 is always selected as a selected green value 260.

Fig 2F illustrates another alternative for selecting an 8 bit value in a red selection diagram 270. In this alternative the red instance 236 is always selected as a selected red value 280.

If the video signal being digitized is grayscale, the three color components will have the same values. For example in Fig 2G, a grayscale pixel 290 comprises a grayscale blue 292 with a value of decimal 40, a grayscale green 294 with a value of 40, and a grayscale red with a value of 40. Because the values are all the same, it makes no difference which grayscale color component is selected, a selected grayscale value 298 will have the value of 40 in this example.

The preferred embodiment of this invention uses the low order byte of the pixel value, which is typically the blue component as shown in Fig 2D.

Fig 2H illustrates a filtered pixel value 299 of 8 bits that may be selected by one of the alternatives described above. In these examples, the filtered pixel value 299 is equivalent to items referenced by numerals 228, 240, 260, 280, or 298. This reduction of the 32 bit pixel value 200 or the 24 bit pixel value 210 contributes a reduction in data size of 4:1 or 3:1, respectively. This reduction recognizes that for some images, such as medical images or grayscale images, no relevant information is lost.

For additional compression, the filtered pixel value 299 can variably select any number of bits. For example, selection of the most significant four bits instead of all eight bits filters noise that may show up in the low order bits may be very suitable for an image such as one produced by an ultrasound medical device. An example of this is shown by ZL4 804 in Fig 8A.

Figs 3A and 3B–Encoding Formats

Speed of compression and decompression may be enhanced if the algorithms fit into computer memory native storage elements such as 8 bit bytes, 16 bit words, or 32 bit double words, or some other size for which the computer architecture is optimized.

A grayscale image may be stored at a higher bit level than the actual values require. This may occur when an image is generated by an imaging technology such as radar, ultrasound, x-ray, magnetic resonance, or similar electronic technology. For example an ultrasound machine may only produce 16 levels of grayscale, requiring 4 bits of data per pixel, but the image digitizing may be performed at 8 to 12 bits per pixel. In this example, the low order bits (4 to 8) respectively provide no significant image data.

In the present invention, a fast and efficient compression and encoding ~~algorithm~~method is implemented by using unused bits to store a repeat count for repeated values.

The most significant N bits of the pixel value are selected where N is the number of significant bits (determined by data analysis or by user selection). If N is less than W, where W is a native machine data type such as 8 bit byte, 16 bit word, or 32 bit double word or some other size for which the computer architecture is optimized, then W-N equals the number of unneeded bits, U. A repeat count, C, can contain a value from 1 to CMAX where CMA is 2 to the power of U. For example, if U equals 4, C can be a number from 1 to 16. In practice the maximum value will be encoded as a zero because the high order bit is truncated. In the example, decimal 16 has a binary value "10000" will be stored as "0000".

For example, when W is 8, value pairs for N and U could include without limitation (2,6), (3,5), (4,4), (5,3), and (6,2). When W is 16, value pairs for N and U could include without limitation (2,14), (3,13), (4,12), (5,11), (6,10), (7, 9), (8, 8), (9, 7), (10, 6), (11, 5), (12, 4), (13, 3), and (14, 2). When W is 32, value pairs for N and U could include without limitation all combinations of values pairs for N and U where N+U equals 32 and $N > 1$ and $U > 1$. When W is not a multiple of 8, value pairs for N and U could include without limitation all combinations of values pairs for N and U where N+U equals W and $N > 1$ and $U > 1$.

Figs 3A shows the encoded format where N 300 represent the N most significant bits of the pixel value 299, U 301 represents the bits that are not used for the data and are

~~therefor~~ used for the repeat count, and W 302 where W is the width of the encoded data and equal to sum of N and U. ~~As stated above W is preferably a native machine element.~~

Fig 3B illustrates bit sub-sampling where N's 300 bit width is 5, U's 301 bit width is 3, and W 302 is 8. The high order 5 bits 310-318 of an 8 bit pixel 325 are extracted to form a five bit sample 330. The lower 3 bits of 330 are ignored bits 350. In the formatted code 360, the ignored bits 350 are replaced with the repeat count value 380.

Encoding

The most significant N bits of each pixel are selected from the image to obtain value V.

In the encryption embodiment of this invention V may be used to select an encoded value, E, from the encoding table. E is also a N-bit value. The number of elements in the encode table 1100 (Fig 11) is 2 to the Nth power.

In the other embodiments of this invention V is used as E.

E is saved as the prior value, P. For each subsequent pixel, the encoded value, E, is obtained and compared to the prior value, P. If the prior value, P, is the same as E, then a repeat counter, C, is incremented; otherwise the accumulated repeat count, C, for the prior value, P, is merged with P and placed in an array A that implements the encoded data 140 (Fig 1) buffer. For example, if W is 8 and N is 4 and C is 10, U is 4, CMAX is 16, and ((P << U) | C) is the merged value. If the repeat count, C, is greater CMAX, then CMAX is merged with P ((P << U) | CMAX) and placed in the encoded data 140 (Fig 1) buffer, A. CMAX is subtracted from C and merged values are placed in A until C is less than CMAX. All pixels are processed in this manner until the final value is compressed and encoded. The length, L, of the encoded data 140 (Fig 1) is ~~recorded with~~also placed in the encoded data 140 buffer.

Fig 4A–Encode Flowchart

Fig 4A illustrates the encode flowchart 400 which represents the details of the encryption embodiment of the encoding step 130 (Fig 1) for the present invention.

The encoding begins at an encode entry 402. In a encode initialization step 403, a prior value P is set to a known value, preferably decimal "255" or hexadecimal 0xFF, a repeat counter C is set to zero, an encoded length L is set to 0, and a completion flag "Done" is set to a logical value of false. Next, a get pixel step 404 obtains a pixel from the image

being encoded. At a get value step 405, a value V is set to the N bit ~~filter~~ filtered pixel value 299 as derived from the pixel using one of the methods shown in Fig 2C to 2G, preferably the fastest as explained above, and extracting the N most significant bits. At a lookup encoded value step 406, an encoded value E is set to the value of one of the codes 1105 (Fig 11A) of the encode table 1100 as indexed by V. (In the non encrypted embodiment of this invention, step 406 is bypassed because V is used as E) Next, a compare previous 408 decision is made by comparing the values of E and P. If the values are the same, ~~a~~ an increment counter step 410 is executed and flow continues to the get pixel step 404 that obtains the next pixel from the image.

If the encode value E does not match the prior value P, then a check count overflow 412 decision is made. If the counter C is less than or equal to CMAX, then a new code step 414 is executed, otherwise a counter overflow step 420 is executed.

At step 414, the counter C is masked and bit-wise OR-ed with P shifted left by U bit positions and is placed in the A at the next available location as indexed by the encoded length L. Then, continuing inside flowchart step 414, L is incremented, the repeat count C is set to 1 and the prior value P is set to E. After step 414, a "check end of data" decision is made by checking to see if there are any more pixels in the image, and, if not, if the last value is has been processed. Because this method utilizes a read ahead technique step 414 must be executed one more time after the end of data is reached to process the last run-length. If there is more data in the image, flow continues to a check of the completion flag "Done" at step 422. If the check indicates that the process is not completed, flow continues to step 404.

If the end of data is reached but the completion flag "Done" is still false, flow continues to a set done step 418. At step 418, the completion flag "Done" is set to logical true, and flow continues to decision 412 where the last run-length will be output and flow will eventually exit through step 414, decision 416, decision 422, and then terminate at encode exit 428.

It is possible for the repeat count C to become larger than CMAX requiring more bits than allocated by this method. This situation is handled by making the check count overflow 412 decision and executing the counter overflow step 420. At step 420, the counter C is masked and bit-wise OR-ed with P shifted left by U bit positions and is placed in the A at the next available location as indexed by the encoded length L. Then, continuing

inside flowchart step 414, L is incremented, the repeat count C is decrement by CMAX. After step 420, flows continues to the check count overflow 412 decision. Thus when the encode value E repeats more that CMAX times, multiple sets of repeat counts and encoded values are output to the encoded data 140 buffer.

This entire process is repeated for each image or video frame selected during optional image sub-sampling (see 110 in Fig 1) and the encoded length L is transmitted with the encoded data associated with each frame. The encoded length varies from frame to frame depending on the content of the image being encoded.

Fig 4B–Image and Pixel Stream

Fig 4B illustrates an image and its corresponding stream of pixels. A rectangular image 430 is composed of rows and columns of pixels. The image 430 has a width 440 and a height 450, both measured in pixels. In this illustrative embodiment, pixels~~Pixels~~ in a row are accessed from left to right. Rows are accessed from top to bottom. Some pixels in the image are labeled from A to Z. Pixel A is the first pixel and pixel Z is the last pixel. Scanning left to right and top to bottom will produce a pixel stream 460. In the pixel stream 460, pixels A and B are adjacent. Also pixels N and O are adjacent even though they appear on different rows in the image. If adjacent pixels have the same code the process in Fig 4A will consider them in the same run.

Because the video signal being digitized is analog there will be some loss of information in the analog to digital conversion. The video digitizing hardware can be configured to sample the analog data into the image 430 with almost any width 440 and any height 450. The present invention achieves most of its effective compression by sub-sampling the data image with the width 440 value less than the conventional 640 and the height 450 value less than the convention 480. ~~The~~ In a preferred embodiment of the invention, for use in a medical application with T1 Internet transmission bandwidth, ~~is to sample~~image dimensions are sub-sampled at 320 by 240. However a image dimension sub-sampling resolution of 80 by 60 may be suitable for some video application.

Figs 5A to 5C–Run-length Encoding Formats of the RHN ~~Algorithm~~Method

Figs 5A to 5C show use of a different structure than the present invention. Figs 5A to 5C show the formats for the run-length encoding of ~~the~~ RHN. In Fig 5A, a code byte 500, with its high order bit designated as a flag bit 510.

Fig 5B shows a repeat code 520 comprising a Boolean value one in its flag bit 510 and a 7 bit count 530 in the remaining 7 low order bits. The seven bit count 530 can represent 128 values with a zero representing "128" and 1 through 127 being their own value.

Fig 5C shows a data code 550 comprising:

1. a Boolean value zero in its flag bit 510
2. two unused data bits: data bit 6 reference by 565 and data bit 5 reference by 570, and
3. five bits, data bits 4 to 0, reference by 575, 580, 585, 590, and 595, respectively.

Fig 5C shows that in every byte of the RHN data code 550 two bits are unused and one bit is used for the flag bit, so that only five of the eight bits are used for data. The remaining three bits are wasted bits 560. The present invention uses a different structure by placing the repeat count in bits that the RHN format would not have used for data (U). The corresponding ZLN format, ZL5 (where N is 5, U is 3, and W is 8), always uses five bits for data and the remaining 3 bits for the repeat count. In practice, repeat counts are small and often can fit in 3 bits, so this embodiment of the present invention will result in superior compression performance over the RHN ~~algorithm~~method.

In addition, the present invention provides for a larger count when the bit filtering is larger. For example, the alternate ZLN format where each byte contains 4 data bits, ZL4 (where N is 4 and U is 4), allows for a four bits of repeat count. For example, in practice, ZL4 is superior to RHN on a typical ultrasound image containing 16 shades of gray.

Fig 6–Encoded Data Stream

Fig 6 shows a series of decimal values 610 comprising a first value 620 equal to decimal 0, a second value 622 equal to 0, a third value 624 equal to 0, a fourth value 626 equal to 0, a fifth value 628 equal to 0, a sixth value 630 equal to 2, and a seventh value 632 equal to 10. After the encoding step 130 (Fig 1), the corresponding encoded data 140 (Fig 1) would be compressed down to three bytes of binary code 640 comprising a first byte 650, a second byte 653, and a third byte 656 each containing a merged value and count, (651, 652), (654, 655), and (657, 658), respectively. The first data 651 has a binary value of "00000" which equals the repeated decimal value zero. The first count 652 has a binary

value "101" which equals decimal five representing the run-length of the repeating value in the first five of the decimal values 610. The second data 654 has a binary value of "00010" which equals the non-repeated decimal value two. The second count 655 has a value of 1. The third data 657 has a binary value of "01010" which equals the non-repeated decimal value ten. The third count 658 has a value of 1.

Fig 7–RHN Codes and Encoded Stream

Fig 7 shows the same series of decimal values 610 (Fig 6) comprising the first value 620 equal to decimal 0, the second value 622 equal to 0, the third value 624 equal to 0, the fourth value 626 equal to 0, the fifth value 728 equal to 0, the sixth value 730 equal to 2, and the seventh value 732 equal to 10. After encoding by RHN, the corresponding encoded data 140 (Fig 1) would be compressed down to four bytes of RHN binary code 740.

The embodiment of the present invention shown in Fig 6 only requires three bytes to encode the same data. In this example, the present invention is 25% better than the ~~ZHN~~ RHN format.

Figs 8A and 8B–ZLN Formats

The ZLN ~~algorithm~~method of the present invention provides for variable formats. The values of N 300, U 301, and W 302 can by dynamically changed between frames. For ease of communication a format is named with the prefix "ZL" and a digit representing the value of N. For example, "ZL5" refers to a format where bit width of N is equal to 5. There are multiple values of U depending of the W. To also specify the bit width of U a hyphen and a number can be appended. For example, "ZL5-13" represents a format where N = 5 and U = 13. ~~"ZL5-8"~~ "ZL5-3" is a common format and may be imprecisely referred to as "ZL5."

Figure 8A shows a number of formats with adjacent labels: ZL3 803, ZL4 804, ZL5 805, ZL8 808, ZL9 809, and ZL12 812. Data bits are represented by "D," and count bits are represented by "C".

Figure 8B shows how the most significant 3 bits of each color component (216, 214, and 212 of Fig 2B) are extracted and formatted in ZL9-7C format (the "C" append indicates that the color is preserved). With three red bits represented by "R", three green bits represented "G" and three blue bits represented by "B".

Decoding

To decode the compressed array, the decoder has a decode table that corresponds with the encode table. For W*4 bit color pixels, the decode table contains the appropriate alpha, red, green, and blue values. For W*3 bit color pixels, the alpha value is not used. The compressed array is processed W bits at a time as X. The repeat count, C, is extracted from X by masking off the data value (C = X & (((2**N)-1) << U)). The encoded value, E, is extracted from X by masking off the count (E = X & ((2**U)-1)). The encoded value, E maybe used to index into the decryption. The decoded pixels are placed in a reconstructed image and repeated C times. Each element of the compressed array, A, is processed until its entire length, L, has been processed.

Fig 9–Decode Flowchart

Fig 9 illustrates the decode flowchart which presents the details of the decryption embodiment of the decode step 160 (Fig 1) and the image reconstitution step 180 (Fig 1).

The decoding beings at a decode entry 900. In a "decode initialization" step 901, a repeat counter C is set to one, an encoded length L is set to the value obtained with the encoded data 140 (Fig 1), and an index I is set to 0. Next, a "get code" step 902 obtains a signed byte X from the encoded data 140 (Fig 1) array A. The index I is incremented. The count ~~#XX (Fig 3X)~~(for example the 3-bit count 380 as shown in Fig 3B) is extracted from X by masking off the data bits and placed in the repeat counter C (C= X & ((2**N)-1 << U). The value of E is extracted from X by masking off the count bits (E = X & (2**U)-1). In practice, the count mask and value mask can be pre-computed with the following two lines of code in the C programming language:

```
valueMask = -1 << U;
countMask = ~valueMask;
```

In ~~the~~ this illustrative decryption embodiment of the present invention, flow goes to a "decode lookup" step 908 where the value of E is used to index into the decode table 1110 (Fig 11) to obtain a pixel value V. In the other embodiments where E is not encrypted, E is used as V and step 908 is bypassed. Flow continues to a "check zero count" 909 decision.

The 909 decision always fails the first time ensuring that a place pixel step 910 is executed. The place pixel step 910 places the pixel value V in the next location of the decompressed image and decrements the repeat counter C and returns to the 909 decision.

The pixel value V is placed repeatedly until C decrements to zero. Then the 909 decision branches flow to a "reset counter" step 914. At step 914 the repeat counter is reset to 1.

Flow continues to the "check length" 916 decision where the index I is compared to the encoded length L to determine if there are more codes to be processed. If I is less than L flow returns to step 902, otherwise the decode process terminates at a "decode exit" 918.

The entire decode process is repeated for each encoded frame image.

Fig 10–Interpolation

Fig 10 interpolation when a two adjacent pixels 1010 and 1012 and two subsequent row adjacent pixels 1014 and 1016 are stretched to insert a new row and column of pixels.

Pixels 1052, 1054, 1056, 1058 and 1060 are ~~insert~~ inserted due to the enlargement of the image. Their values are calculated by averaging the values of the two pixels above and below or to the left or the right of the new pixel. A preferred sequence is calculation of:

1. 1052 between 1010 and 1012
2. 1054 between 1010 and 1014
3. 1058 between 1012 and 1016
4. 1056 between 1054 and 1058

Pixel ~~160~~ 1060 can be calculated on the interpolation for the subsequent row.

Fig 11–Encryption

By using corresponding encoding and decoding tables the data can be encrypted and decrypted without using actual values. Encryption provides a level of security for the encoded data 140 while in storage or transit.

Fig 11 shows an example of an encryption table 1100, where N is 3 and W is 8, and a decryption table 1110, where N is 3 and U is 5.

The encode table 1100 is 2 the power of N in length. If the target color image format is W*4 bit color, then the decode table 1110 has W bits for alpha, red, green, and blue each, respectively. If the target color image format is W*3 bit color, then the alpha value is not used. If the image is W bit grayscale then only the grayscale value is used to create the decompressed and decoded image.

The corresponding table elements are mapped to each other. For example, 0 could encode to 22 as long as the 22nd element of the decode table returns (`0xff << 24 | 0 << 16 | 0 << 8 | 0`).

When these versions of the tables are used, the encode and decode processes and their speed of execution are substantially the same but the encoded data 140 (Fig 1) becomes a cipher and has a higher level of security. It should be recognized by one with ordinarily skill in the art that there are other embodiments of the present invention with different encryption/decryption table rearrangements.

Advantages

Noise Filtering and Image Enhancement

The removal of the least significant bits of pixel values results in high quality decompressed images when the original image is generated by an electronic sensing device such as an ultrasound machine which is generating only a certain number of bits of grayscale resolution. By variably altering the number of most significant bits various filters can be implemented to enhance the image quality. Such a noise filter can be beneficial when the image is generated by an imaging technology such as radar, ultrasound, x-ray, magnetic resonance, or similar technology. Variations can be made to enhance the perceived quality of the decompressed image. Therefore, altering the number of data bits selected and altering the width of the repeat count is anticipated by this invention and specific values in the examples should not be construed as limiting the scope of this invention.

Dynamic Variable Formats

While a video stream is being viewed a viewer on the decoding end of the transmission can vary the settings for the compressor. Different tradeoffs between image spatial and temporal quality can be made. As the contents of the video signal change an appropriate format can be selected. Control signals can be sent back to the compressor via a communications link.

Execution Speed

The preferred embodiment of this invention use a number of techniques to reduce the time required to compress and decompress the data.

The methods require only a single sequential pass through the data. Both the compression steps 100 and the decompression steps 150 access a pixel once and perform all calculations.

When selecting the filtered pixel value 299, the preferred embodiment selects the low order byte from the 32 bit pixel value 200 or the 24 bit pixel value 210 so that an additional shift operation or addressing operation is avoided.

The shift operation is a fast and efficient way to convert a byte or word to the filtered pixel value 299.

General Purpose

The lossless compression of the sampled data achieved by the preferred embodiment of the present invention results in high quality video streams that have general purpose application in a number of areas including, without limitation, video conferencing, surveillance, manufacturing, rich media advertising, and other forms of video transmission, storage, and processing.

Lossless Nature/No Artifacts

Once the analog signal is sub-sampled and filtered to select a filtered pixel value which eliminates some of the real world defects, the methods of the present invention compress and decompress the data with no irreversible data loss. Unlike JPEG and MPEG, the decompressed image never suffers from artificially induced blocking or smearing or other artifacts that are result of the lossy compression algorithm itself. As a result even a small sub-sample of the image remains clear and true to the perceived quality of the original image.

Superior Features over RHN Format

When compared against the RHN format, the format and methods of the present invention provide a number of advantages, including, but not limited to, faster speed and smaller size of encoded data, better performance for both medical and typical video images, and a typically closer representation of the original video signal.

Conclusion, Ramification, and Scope

Accordingly, the reader will see that the compression and decompression steps of the present invention provides a means of digitally compressing a video signal in real time,

communicating the encoded data stream over a transmission channel, and decoding each frame and displaying the decompressed video frames in real time.

Furthermore, the present invention has additional advantages in that:

1. it provides a means of filtering real world defects from the video image and enhancing the image quality;
2. it allows for execution of both the compression and decompression steps using software running on commonly available computers without special compression or decompression hardware;
3. it ~~provide~~ provides decompressed images that have high spatial quality that are not distorted by artifacts of the compression algorithms being used;
4. ~~is~~ it provides a variably scalable means of video compression; and
5. it provides a means for reducing the space required in a storage medium.

Although the descriptions above contain many specifics, these should not be construed as limiting the scope of the invention but as merely providing illustrations of some of the preferred embodiments of this invention. For example, bit ordering can be altered and the same relative operation, relative performance, and relative perceived image quality will result. Also, these processes can each be implemented as a hardware apparatus that will improve the performance significantly.

Thus the scope of the invention should be determined by the appended claims and their legal equivalents, and not solely by the examples given.

Claims: I claim:

1. A method of compression of graphic images which make up a video stream, comprising the steps of:
   (a) sub-sampling a number of pixel bits from an image selected from said graphic images;
   (b) run-length encoding repeated instances of said number of pixel bits;
   repeating steps ~~(b) and (c)~~(a) and (b) until each said number of pixel bits is encoded in an encoded data buffer.
2. The method of claim 1 wherein the image dimensions are less than or equal to 320 by 240.
3. The method of claim 1 wherein said number of pixel bits is one of the set of 3, 4, 5, 8, 9, 12, 15, 16, and 24.
4. The method of claim 3 wherein said number of pixel bits is extracted from the most significant bits of each color component.
5. An encoded video signal comprising a series of said encoded data buffers, wherein said data buffers were encoded according to the method of claim 1.
6. A storage medium in which the encoded video signal as claimed in claim 5 is stored.
7. A method of decompressing an encoded video signal, comprising the steps of:
   (a) reading a stream of run-length encoded codes;
   (b) determining a series of pixels based on the values and run-lengths of said codes;
   (c) combining said pixels into an image~~; and~~.
8. The method of claim 7 further comprising the step of displaying a series of said images.
9. The method of claim 7 wherein the width and the height of said image are less than or equal to 320 by 240, respectively.
10. The method of claim 7 wherein said codes represent the combination most significant bits of each of the color components of each pixel.
11. A machine for compressing of a plurality of video frames which make up a video signal, comprising:
   (a) a video digitizer configured to digitizing a frame from said video frames;

(b) a video memory which is able to receive a plurality of pixels from said video digitizer;

(c) ~~a~~ an encoding circuit for counting repeated instances of a pixel value when scanning said plurality of pixels and outputting a series of encoded data comprising a combined run-length field and a data field.

(d) a memory which is able to store said encoded data;

(e) an input/output device.

12. The machine of claim 11 wherein said encoding circuit variably selects one of a set of 3, 4, 5, 8, 9, 12, 15, 16, and 24, as the number of pixel bits.

13. The machine of claim 12 wherein said pixel value is extracted from the most significant bits of each color component.

14. The machine of claim 11 wherein said input/output device is a storage medium.

15. The machine of claim 11 wherein said input/output device is a communications transmission channel.

16. A machine for decompressing an stream of encoded data that represents a video signal, comprising:

(a) an input/output device for reading said stream of encoded data;

(b) a decoding circuit which can decode the encoded data and output a stream of pixel values; and

(c) a memory that is able to store an image comprising said stream of pixel values that can be displayed as frames of a video sequence.

17. The method of claim 1 wherein one or more of the settings of width, height, frame rate, brightness, and contrast of said images are variably altered by a receiver of said encoded data.

18. The method of claim 1 wherein said number of pixel bits are variably altered by a receiver of said encoded data.

19. The method of claim 1 further comprising a step of compressing said buffer with a lossless technique known in the art.

20. The method of claim 8 wherein said images are enlarged by stretching prior to said displaying.

21. The method of claim 1 further comprising a step of encrypting said number of pixel bits.

22. The method of claim 1 wherein said graphic images have a first predetermined frame rate and a subset of said graphic images are sub-sampled at a second frame rate that was less than the first frame rate such that only a subset of said graphic images are selected from the original set of said graphic images, and
wherein said image selected from said graphic images is a sub-sampled image such that it is one of said subset of sub-sampled images.

23. The method of claim 1 wherein the image dimensions of said video stream is greater than 320 pixels wide and 240 pixels high, and
wherein said method further comprises the step of first dimensionally sub-sampling an image from said graphic images such that the sub-sampled image dimensions of said image are less than or equal to 320 by 240.

24. The method of claim 1 wherein a length of the encoded data in said encoded data buffer in placed in said encoded data buffer.

25. The method of claim 7 further comprising the step of reading a length of the encoded data and using said length to determine when all the encoded data has been processed.

VARIABLY GENERAL PURPOSE COMPRESSION FOR VIDEO IMAGES (ZLN)

Abstract: Methods, medium, and machines which compress, enhance, encode, transmit, decompress and display digital video images in real time. Real time compression is achieved by sub-sampling each frame of a video signal, filtering the pixel values, and encoding. Real time transmission is achieved due to high levels of effective compression. Real time decompression is achieved by decoding and decompressing the encoded data to display high quality images. Receiver can alter various setting including but not limited to the format for the compression, image size, frame rate, brightness and contrast.

Application # 09/467,721

CLEAN COPY WITH ALL PENDING CLAIMS

#6 RE
BA 4/9/03

Patent Application of

Kendyl A. Román

for

TITLE: VARIABLE GENERAL PURPOSE COMPRESSION FOR VIDEO IMAGES (ZLN)

CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims priority under 35 U.S.C. § 119(e) of the co-pending U.S. provisional application Serial Number 60/113,051 filed on 1998 December 21, and entitled "METHODS OF ZERO LOSS (ZL) COMPRESSION AND ENCODING OF GRAYSCALE IMAGES." The provisional application Serial Number 60/113,051 filed on 1998 December 21 and entitled "METHODS OF ZERO LOSS (ZL) COMPRESSION AND ENCODING OF GRAYSCALE IMAGES" is also hereby incorporated by reference.

A U.S. patent application Serial Number 09/470,566, filed on 1999 December 22, and entitled GENERAL PURPOSE COMPRESSION FOR VIDEO IMAGES (RHN)", to be known as the "RHN" method, has claims that combine some of the elements of the present invention in a different combination. The RHN application claims a priority date based on a co-pending U.S. provisional application Serial Number 60/113,276 filed on 1998 December 23. The provisional application Serial Number 60/113,276 filed on 1998 December 23 is also hereby incorporated by reference. The application Serial Number 09/470,566, filed on 1999 December 22, and entitled GENERAL PURPOSE COMPRESSION FOR VIDEO IMAGES (RHN)" as amended is also hereby incorporated by reference.

My co-pending U.S. patent application Serial Number 90/312,922 filed on 1999 May 17, and entitled "SYSTEM FOR TRANSMITTING VIDEO IMAGES OVER A COMPUTER NETWORK TO A REMOTE RECEIVER" describes an embodiment of the

invention of the RHN method, as well as a system for practicing the compression method. U.S. patent application Serial Number 90/312,922, filed on 1999 May 17, and entitled "SYSTEM FOR TRANSMITTING VIDEO IMAGES OVER A COMPUTER NETWORK TO A REMOTE RECEIVER" is also hereby incorporated by reference.

My co-pending U.S. patent application, serial number _____, filed on 1999 December 20, and entitled "ADDING DOPPLER ENHANCEMENT TO GRAYSCALE COMPRESSION (ZLD)" describes an invention that is related to this application. U.S. patent application Serial Number _____, filed on 1999 December 20, and entitled "ADDING DOPPLER ENHANCEMENT TO GRAYSCALE COMPRESSION (ZLD)" as amended is also hereby incorporated by reference.

BACKGROUND – FIELD OF THE INVENTION

This invention relates to data compression, specifically to the compression and decompression of video images.

BACKGROUND–DESCRIPTION OF PRIOR ART

In the last few years, there have been tremendous advances in the speed of computer processors and in the availability of bandwidth of worldwide computer networks such as the Internet. These advances have led to a point where businesses and households now commonly have both the computing power and network connectivity necessary to have point-to-point digital communications of audio, rich graphical images, and video. However the transmission of video signals with the full resolution and quality of television is still out of reach. In order to achieve an acceptable level of video quality, the video signal must be compressed significantly without losing either spatial or temporal quality.

A number of different approaches have been taken but each has resulted in less than acceptable results. These approaches and their disadvantages are disclosed by Mark Nelson in a book entitled The Data Compression Book, Second Edition, published by M&T Book in 1996. Mark Morrision also discusses the state of the art in a book entitled The Magic of Image Processing, published by Sams Publishing in 1993.

Video Signals

Standard video signals are analog in nature. In the United States, television signals contain 525 scan lines of which 480 lines are visible on most televisions. The video signal represents a continuous stream of still images, also known as frames, that are fully scanned, transmitted and displayed at a rate of 30 frames per second. This frame rate is considered full motion.

A television screen has a 4:3 aspect ratio.

When an analog video signal is digitized each of the 480 lines are sampled 640 times, and each sample is represented by a number. Each sample point is called a picture element, or pixel. A two dimensional array is created that is 640 pixels wide and 480 pixels high. This 640 x 480 pixel array is a still graphical image that is considered to be full frame. The human eye can perceive 16.7 thousand colors. A pixel value comprised of 24 bits can represent each perceivable color. A graphical image made up of 24-bit pixels is considered

to be full color. A single, second-long, full frame, full color video requires over 220 millions bits of data.

The transmission of 640 x 480 pixels x 24 bits per pixel times 30 frames requires the transmission of 221,184,000 millions bits per second. A T1 Internet connection can transfer up to 1.54 millions bits per second. A high speed (56Kb) modem can transfer data at a maximum rate of 56 thousand bits per second. The transfer of full motion, full frame, full color digital video over a T1 Internet connection, or 56Kb modem, will require an effective data compression of over 144:1, or 3949:1, respectively.

A video signal typically will contain some signal noise. In the case where the image is generated based on sampled data, such as an ultrasound machine, there is often noise and artificial spikes in the signal. A video signal recorded on magnetic tape may have fluctuations due the irregularities in the recording media. Florescent or improper lighting may cause a solid background to flicker or appear grainy. Such noise exists in the real world but may reduce the quality of the perceived image and lower the compression ratio that could be achieved by conventional methods.

Basic Run-length Encoding

An early technique for data compression is run-length encoding where a repeated series of items are replaced with one sample item and a count for the number of times the sample repeats. Prior art shows run-length encoding of both individual bits and bytes. These simple approaches by themselves have failed to achieve the necessary compression ratios.

Variable Length Encoding

In the late 1940s, Claude Shannon at Bell Labs and R.M. Fano at MIT pioneered the field of data compression. Their work resulted in a technique of using variable length codes where codes with low probabilities have more bits, and codes with higher probabilities have fewer bits. This approach requires multiple passes through the data to determine code probability and then to encode the data. This approach also has failed to achieve the necessary compression ratios.

D. A. Huffman disclosed a more efficient approach of variable length encoding known as Huffman coding in a paper entitled "A Method for Construction of Minimum

Redundancy Codes," published in 1952. This approach also has failed to achieve the necessary compression ratios.

Arithmetic, Finite Context, and Adaptive Coding

In the 1980s, arithmetic, finite coding, and adaptive coding have provided a slight improvement over the earlier methods. These approaches require extensive computer processing and have failed to achieve the necessary compression ratios.

Dictionary-Based Compression

Dictionary-based compression uses a completely different method to compress data. Variable length strings of symbols are encoded as single tokens. The tokens form an index to a dictionary. In 1977, Abraham Lempel and Jacob Ziv published a paper entitled, "A Universal Algorithm for Sequential Data Compression" in IEEE Transactions on Information Theory, which disclosed a compression technique commonly known as LZ77. The same authors published a 1978 sequel entitled, "Compression of Individual Sequences via Variable-Rate Coding," which disclosed a compression technique commonly known as LZ78 (see U.S. Patent 4,464,650). Terry Welch published an article entitled, "A Technique for High-Performance Data Compression," in the June 1984 issue of IEEE Computer, which disclosed an algorithm commonly known as LZW, which is the basis for the GIF algorithm (see US Patents 4,558,302, 4,814,746, and 4,876,541). In 1989, Stack Electronics implemented a LZ77 based method called QIC-122 (see U.S. Patent 5,532,694, U.S. Patent 5,506,580, and U.S. Patent 5,463,390).

These lossless (method where no data is lost) compression methods can achieve up to 10:1 compression ratios on graphic images typical of a video image. While these dictionary-based algorithms are popular, these approaches require extensive computer processing and have failed to achieve the necessary compression ratios.

JPEG and MPEG

Graphical images have an advantage over conventional computer data files: they can be slightly modified during the compression/decompression cycle without affecting the perceived quality on the part of the viewer. By allowing some loss of data, compression ratios of 25:1 have been achieved without major degradation of the perceived image. The Joint Photographic Experts Group (JPEG) has developed a standard for graphical image compression. The JPEG lossy (method where some data is lost) compression algorithm first

divides the color image into three color planes and divides each plane into 8 by 8 blocks, and then the algorithm operates in three successive stages:

(a) A mathematical transformation known as Discrete Cosine Transform (DCT) takes a set of points from the spatial domain and transforms them into an identical representation in the frequency domain.

(b) A lossy quantization is performed using a quantization matrix to reduce the precision of the coefficients.

(c) The zero values are encoded in a zig-zag sequence (see Nelson, pp. 341-342).

JPEG can be scaled to perform higher compression ratio by allowing more loss in the quantization stage of the compression. However this loss results in certain blocks of the image being compressed such that areas of the image have a blocky appearance and the edges of the 8 by 8 blocks become apparent because they no longer match the colors of their adjacent blocks. Another disadvantage of JPEG is smearing. The true edges in an image get blurred due to the lossy compression method.

The Moving Pictures Expert Group (MPEG) uses a combination of JPEG based techniques combined with forward and reverse temporal differencing. MPEG compares adjacent frames and for those blocks that are identical to those in a previous or subsequent frame and only a description of the previous or subsequent identical block is encoded. MPEG suffers from the same blocking and smearing problems as JPEG.

These approaches require extensive computer processing and have failed to achieve the necessary compression ratios without unacceptable loss of image quality and artificially induced distortion.

QuickTime: CinePak, Sorensen, H.263

Apple Computer, Inc. released a component architecture for digital video compression and decompression, named QuickTime. Any number of methods can be encoded into a QuickTime compressor/decompressor (codec). Some popular codec are CinePak, Sorensen, and H.263. CinePak and Sorensen both require extensive computer processing to prepare a digital video sequence for playback in real time; neither can be used for live compression. H.263 compresses in real time but does so by sacrificing image quality resulting in severe blocking and smearing.

Fractal and Wavelet Compression

Extremely high compression ratios are achievable with fractal and wavelet compression algorithms. These approaches require extensive computer processing and generally cannot be completed in real time.

Sub-sampling

Sub-sampling is the selection of a subset of data from a larger set of data. For example, when every other pixel of every other row of a video image is selected, the resulting image has half the width and half the height.

Image Stretching

If an image is to be enlarged but maintain the same number of pixels per inch, data must be filled in the new pixels that are added. Various methods of stretching an imaging and filling in the new pixels to maintain image consistency. Some methods known in the art are dithering (using adjacent colors that appear to be blended color), and error diffusion, "nearest neighbor", bilinear and bicubic.

SUMMARY OF THE INVENTION

In accordance with the present invention a method of compression of a video stream comprises steps of sub-sampling a video frame, and run-length encoding the sub-sampled pixel values, whereby the method can be executed in real time and the compressed representation of pixels saves substantial space on a storage medium and require substantially less time and bandwidth to be transported over a communications link. The present invention includes a corresponding method for decompressing the encoded data.

Objects and Advantages

Accordingly, beside the objects and advantages of the method described in our patent above, some additional objects and advantages of the present invention are:

(a) to provide a method of compressing and decompressing video signals so that the video information can be transported across a digital communications channel in real time.

(b) to provide a method of compressing and decompressing video signals such that compression can be accomplished with software on commercially available computers without the need for additional hardware for either compression or decompression.

(c) to provide a high quality video image without the blocking and smearing defects associated with prior art lossy methods.

(d) to provide a high quality video image that suitable for use in medical applications.

(e) to enhance images by filtering noise or recording artifacts.

(f) to provide a method of compression of video signals such that the compressed representation of the video signals is substantially reduced in size for storage on a storage medium.

(g) to provide a level of encryption so that images are not directly viewable from the data as contained in the transmission.

DRAWING FIGURES

In the drawings, closely related figures have the same number but different alphabetic suffixes.

Fig 1 shows the high level steps of compression and decompression of an image.

Fig 2A to 2H show alternatives for selecting a pixel value for encoding.

Fig 3A shows the variable encoding format.

Fig 3B shows an example of a code where N is 5 bits wide and U is 3 bits wide.

Fig 4A shows the flowchart for the compression method.

Fig 4B shows an image and a corresponding stream of pixels.

Fig 5A to 5C shows the formats for the run-length encoding of the RHN method.

Fig 6 shows a series of codes and the resulting encoded stream.

Fig 7 shows a series of codes and the resulting encoded stream of the RHN method.

Fig 8A shows examples of variable formats.

Fig 8B shows a format that preserves 9 bits of color.

Fig 9 shows the flow chart for the decompression method.

Figs 10 shows image stretching by interpolation.

Figs 11A and 11B show an encryption table and a decryption table.

Figs 12A and 12B show an machines for compressing and decompressing, respectively.

Fig 12C shows a compressor and decompressor connected to a storage medium

Fig 12D shows a compressor and decompressor connected to a communications channel.

Fig 13A shows elements of a compressor.

Fig 13B shows an embodiment of an encoding circuit.

Fig 13C shows a generic pixel sub-sampler.

Figs 13D through 13J show embodiments of pixel sub-samplers.

Figs 14A through 14c shows embodiments of a machine element for variably altering the number of bits.

Fig 15 shows elements of a decompressor.

Fig 16A shows elements for setting width, height, frame rate, brightness, and contrast which are variably altered by a receiver.

Fig 16B shows elements for setting the number of pixel bits which are variably altered by a receiver.

Fig 17 shows a lossless compression step for further compression an encoded data buffer.

Fig 18 shows images being enlarged by stretching.

Reference Numerals in Drawings

100	compression steps	110	sub-sampling step
130	encoding step		
140	encoded data	150	decompression steps
160	decoding step	180	image reconstitution step
200	32 bit pixel value		
202	blue channel	204	green channel
206	red channel	208	alpha channel
210	24 bit pixel value	212	blue component
214	green component	216	red component
220	RGB averaging diagram	222	blue value
224	green value	226	red value
228	averaged value	230	blue selection diagram
232	blue instance	234	green instance

236	red instance	240	selected blue value
250	green selection diagram	260	selected green value
270	red selection diagram	280	selected red value
290	grayscale pixel	292	grayscale blue
294	grayscale green	296	grayscale red
298	selected grayscale value	299	filtered pixel value
300	N	301	U
302	W	310	pixel bit 7
312	pixel bit 6	314	pixel bit 5
316	pixel bit 4	318	pixel bit 3
320	pixel bit 2	322	pixel bit 1
324	pixel bit 0	325	8 bit pixel
330	5 bit sample		
332	sample bit 4	334	sample bit 3
336	sample bit 2	338	sample bit 1
340	sample bit 0	350	3 low order bits
360	formatted code	380	3 bit count value
400	encode flowchart		
402	encode entry	403	encode initialization step
404	get pixel step	405	get value step
406	lookup encoded value step	408	compare previous
410	increment counter step	412	check count overflow
414	new code step	416	check end of data
418	set done	420	counter overflow step
422	check done	428	encode exit
430	image	440	image width
450	image height	460	pixel stream
500	code byte	510	flag bit
520	repeat code	530	count
550	data code	560	wasted bits
565	data bit 6		
570	data bit 5	575	data bit 4

580 data bit 3
585 data bit 2
590 data bit 1
595 data bit 0
610 decimal values
620 first value
622 second value
624 third value
626 fourth value
628 fifth value
630 sixth value
632 seventh value
640 binary code
650 first byte
651 first data
652 first count
653 second byte
654 second data
655 second count
656 third byte
657 third data
658 third count
740 RHN binary code
803 ZL3 format
804 ZL4 format
805 ZL5 format
808 ZL8 format
809 ZL9 format
812 ZL12 format
820 ZL9C format
900 decode entry
901 decode initialize step
902 get code step
908 decode lookup step
909 check zero count
910 place pixel step
914 reset counter step
916 check length
918 decode exit
1000 encryption key
1010 first adjacent pixel
1012 second adjacent pixel
1010 first subsequent adjacent pixel
1012 second subsequent adjacent pixel
1052, 1054, 1056, 1058, 1060 interpolated pixels
1100 encryption table
1110 decryption table
1200 video frames
1205a first video frame
1205b second video frame
1205n nth video frame
1210 compressor
1215 video signal
1220 series of encoded data
1225 encoded data buffer
1225a first encoded data
1225b second encoded data
1225n nth encoded data
1230a first received encoded data

1230b second received encoded data
1230n nth received encoded data
1238 received encoded data
1235 encoded data stream
1240 I/O device
1245 input encoded data stream
1250 decompressor
1260a first decoded video frame
1260b second decoded video frame
1260n nth decoded video frame
1268 decoded video frames
1270 video sequence
1280 storage medium
1290 communications channel
1310 video digitizer
1320 path 1320
1330 video memory
1331 scan
1332 pixel index
1340 path 1340
1350 encoding circuit
1360 path 1360
1370 encoded data
1380 pixel sub-sampler
1380a 24 to 5 bit sub-sampler
1380b 24-bit RGB to 5 bit sub-sampler
1380c 32-bit RGB to 5 bit sub-sampler
1380d color 9-bit sub-sampler
1380e YUV sub-sampler
1380f 36-bit RGB to 24-bit sub-sampler
1380g 15-bit sub-sampler
~~1380 pixel sub-sampler~~
~~1380 pixel sub-sampler~~
~~1380 pixel sub-sampler~~
1382 pixel extractor
1383 value path
1384 coder
1385 path 1385
1390 data/count
1392 code index
1395 path 1395
1400 24-bit to variable bit sub-sampler
1401 generic 3-bit sub-sampler
1402 generic 4-bit sub-sampler
1403 generic 8-bit sub-sampler
1404 generic 10-bit sub-sampler
1410 number of bits selector
1420 number of bits indicator

1430 36-bit to variable bit sub-sampler
1440 24/36 bit variable bit sub-sampler
1450 second selector
1460 selection logic
1510 decoding circuit
1520 decoded pixel values
1530 decoder pixel index
1540 image memory
1600 transmitter
1610 receiver
1615 setting control path
1620 frame sub-sampler
1621 path 1621
1630 selected frame
1632 pixel from frame
1640 transmitter pixel sub-sampler
1642 path 1642
1650 run length encoder
1660 settings
1661 brightness
1662 contract
1663 height
1664 width
1665 frame rate
1670 frame selector
1675 frame select indicator
1680 number of pixel bits setting
1700 run-length encoding step
1710 run-length encoded output
1720 further lossless compression step
1730 further lossless compression output
1800 unstretched frame
1810 enlarged image
1820 stretching step

DESCRIPTION OF THE INVENTION

Fig 1–Compression and Decompression Steps

Fig 1 illustrates a sequence of compression steps 100 and a sequence of decompression steps 150 of the present invention. The compression steps 100 comprise a sub-sampling step 110 and an encoding step 130. After completion of the compression steps 100, a stream of encoded data 140 is output to either a storage medium or a transmission

channel. The decompression steps 150 comprise a decoding step 160 wherein the stream of encoded data 140 is processed and an image reconstitution step 180.

Figs 2A to 2H Selecting Pixel Values for Encoding

Figs 2A to 2G illustrate alternatives for selecting a pixel value for encoding. The sub-sampling step 110 (Fig 1) includes sub-sampling of a pixel value to obtain a variable selected number of bits.

Video digitizing hardware typical has the options of storing the pixel values as a 32 bit pixel value 200 or a 24 bit pixel value 210, shown in Fig 2A and Fig 2B, respectively. The 32 bit pixel value 200 is composed of a blue channel 202, a green channel 204, a red channel 206, and an alpha channel 208. Each channel contains of 8 bits and can represent 256 saturation levels for the particular color channel. For each channel the saturation intensity value of zero represents the fully off state, and the saturation intensity value of "255" represents the fully on state. A common alternative not shown is a sixteen bit format where the three color channels contain 5 bits each and the alpha channel is a single bit. The present invention anticipates the use of the color channels of 16 bit pixel value is a manner substantially the same as the 32-bit pixel value 200 except the number of bits per channel is 5 instead of 8.

The 24 bit pixel value 210 is composed of a blue component 212, a green component 214, and a red component 216. There is no component for the alpha channel in the 24 bit pixel value 210. Regardless of the structure, the blue channel 202 is equivalent to the blue component 212, the green channel 204 is equivalent to the green component 214, and the red channel 206 is equivalent to the red component 216.

In the present invention, the 32 bit pixel value 200 alternative is preferred due to the consistent alignment of 32 bit values in most computer memories; however for simplicity of illustration the alpha channel 208 will be omitted in Fig 2C to 2G.

If the video signal is digitized in color, the three color components may have different values. For example in Fig 2C, a RGB averaging diagram 220 illustrates a blue value 222 of 35 decimal, a green value 224 of 15, and a red value 226 of 10. One alternative is to sub sample from 24 bits to 8 bits by averaging the three color values to obtain an averaged value 228 that, in this example, has the value of 20. (10+15+35)/3 = 20. This will

produce a grayscale image. Alternatively, a color image can be preserved by sampling bits from each color component (see Fig 8B).

Fig 2D illustrates another alternative for selecting an 8 bit value in a blue selection diagram 230. In this example, a blue instance 232 has the value of 35, a green instance 234 has the value of 15, and a red instance 236 has the value of 10. In this alternative the blue instance 232 is always selected as a selected blue value 240.

Fig 2E illustrates another alternative for selecting an 8 bit value in a green selection diagram 250. In this alternative the green instance 234 is always selected as a selected green value 260.

Fig 2F illustrates another alternative for selecting an 8 bit value in a red selection diagram 270. In this alternative the red instance 236 is always selected as a selected red value 280.

If the video signal being digitized is grayscale, the three color components will have the same values. For example in Fig 2G, a grayscale pixel 290 comprises a grayscale blue 292 with a value of decimal 40, a grayscale green 294 with a value of 40, and a grayscale red with a value of 40. Because the values are all the same, it makes no difference which grayscale color component is selected, a selected grayscale value 298 will have the value of 40 in this example.

The preferred embodiment of this invention uses the low order byte of the pixel value, which is typically the blue component as shown in Fig 2D.

Fig 2H illustrates a filtered pixel value 299 of 8 bits that may be selected by one of the alternatives described above. In these examples, the filtered pixel value 299 is equivalent to items referenced by numerals 228, 240, 260, 280, or 298. This reduction of the 32 bit pixel value 200 or the 24 bit pixel value 210 contributes a reduction in data size of 4:1 or 3:1, respectively. This reduction recognizes that for some images, such as medical images or grayscale images, no relevant information is lost.

For additional compression, the filtered pixel value 299 can variably select any number of bits. For example, selection of the most significant four bits instead of all eight bits filters noise that may show up in the low order bits may be very suitable for an image such as one produced by an ultrasound medical device. An example of this is shown by ZL4 804 in Fig 8A.

Figs 3A and 3B–Encoding Formats

Speed of compression and decompression may be enhanced if the algorithms fit into computer memory native storage elements such as 8 bit bytes, 16 bit words, or 32 bit double words, or some other size for which the computer architecture is optimized.

A grayscale image may be stored at a higher bit level than the actual values require. This may occur when an image is generated by an imaging technology such as radar, ultrasound, x-ray, magnetic resonance, or similar electronic technology. For example an ultrasound machine may only produce 16 levels of grayscale, requiring 4 bits of data per pixel, but the image digitizing may be performed at 8 to 12 bits per pixel. In this example, the low order bits (4 to 8) respectively provide no significant image data.

In the present invention, a fast and efficient compression and encoding method is implemented by using unused bits to store a repeat count for repeated values.

The most significant N bits of the pixel value are selected where N is the number of significant bits (determined by data analysis or by user selection). If N is less than W, where W is a native machine data type such as 8 bit byte, 16 bit word, or 32 bit double word or some other size for which the computer architecture is optimized, then W-N equals the number of unneeded bits, U. A repeat count, C, can contain a value from 1 to CMAX where CMA is 2 to the power of U. For example, if U equals 4, C can be a number from 1 to 16. In practice the maximum value will be encoded as a zero because the high order bit is truncated. In the example, decimal 16 has a binary value "10000" will be stored as "0000".

For example, when W is 8, value pairs for N and U could include without limitation (2,6), (3,5), (4,4), (5,3), and (6,2). When W is 16, value pairs for N and U could include without limitation (2,14), (3,13), (4,12), (5,11), (6,10), (7, 9), (8, 8), (9, 7), (10, 6), (11, 5), (12, 4), (13, 3), and (14, 2). When W is 32, value pairs for N and U could include without limitation all combinations of values pairs for N and U where N+U equals 32 and $N > 1$ and $U > 1$. When W is not a multiple of 8, value pairs for N and U could include without limitation all combinations of values pairs for N and U where N+U equals W and $N > 1$ and $U > 1$.

Figs 3A shows the encoded format where N 300 represent the N most significant bits of the pixel value 299, U 301 represents the bits that are not used for the data and are used for the repeat count, and W 302 where W is the width of the encoded data and equal to sum of N and U

Fig 3B illustrates bit sub-sampling where N's 300 bit width is 5, U's 301 bit width is 3, and W 302 is 8. The high order 5 bits 310-318 of an 8 bit pixel 325 are extracted to form a five bit sample 330. The lower 3 bits of 330 are ignored bits 350. In the formatted code 360, the ignored bits 350 are replaced with the repeat count value 380.

Encoding

The most significant N bits of each pixel are selected from the image to obtain value V.

In the encryption embodiment of this invention V may be used to select an encoded value, E, from the encoding table. E is also a N-bit value. The number of elements in the encode table 1100 (Fig 11) is 2 to the Nth power.

In the other embodiments of this invention V is used as E.

E is saved as the prior value, P. For each subsequent pixel, the encoded value, E, is obtained and compared to the prior value, P. If the prior value, P, is the same as E, then a repeat counter, C, is incremented; otherwise the accumulated repeat count, C, for the prior value, P, is merged with P and placed in an array A that implements the encoded data 140 (Fig 1) buffer. For example, if W is 8 and N is 4 and C is 10, U is 4, CMAX is 16, and ((P << U) | C) is the merged value. If the repeat count, C, is greater CMAX, then CMAX is merged with P ((P << U) | CMAX) and placed in the encoded data 140 (Fig 1) buffer, A. CMAX is subtracted from C and merged values are placed in A until C is less than CMAX. All pixels are processed in this manner until the final value is compressed and encoded. The length, L, of the encoded data 140 (Fig 1) is also placed in the encoded data 140 buffer.

Fig 4A–Encode Flowchart

Fig 4A illustrates the encode flowchart 400 which represents the details of the encryption embodiment of the encoding step 130 (Fig 1) for the present invention.

The encoding begins at an encode entry 402. In a encode initialization step 403, a prior value P is set to a known value, preferably decimal "255" or hexadecimal 0xFF, a repeat counter C is set to zero, an encoded length L is set to 0, and a completion flag "Done" is set to a logical value of false. Next, a get pixel step 404 obtains a pixel from the image being encoded. At a get value step 405, a value V is set to the N bit filtered pixel value 299 as derived from the pixel using one of the methods shown in Fig 2C to 2G, preferably the fastest as explained above, and extracting the N most significant bits. At a lookup encoded

value step 406, an encoded value E is set to the value of one of the codes 1105 (Fig 11A) of the encode table 1100 as indexed by V. (In the non encrypted embodiment of this invention, step 406 is bypassed because V is used as E) Next, a compare previous 408 decision is made by comparing the values of E and P. If the values are the same, an increment counter step 410 is executed and flow continues to the get pixel step 404 that obtains the next pixel from the image.

If the encode value E does not match the prior value P, then a check count overflow 412 decision is made. If the counter C is less than or equal to CMAX, then a new code step 414 is executed, otherwise a counter overflow step 420 is executed.

At step 414, the counter C is masked and bit-wise OR-ed with P shifted left by U bit positions and is placed in the A at the next available location as indexed by the encoded length L. Then, continuing inside flowchart step 414, L is incremented, the repeat count C is set to 1 and the prior value P is set to E. After step 414, a "check end of data" decision is made by checking to see if there are any more pixels in the image, and, if not, if the last value is has been processed. Because this method utilizes a read ahead technique step 414 must be executed one more time after the end of data is reached to process the last run-length. If there is more data in the image, flow continues to a check of the completion flag "Done" at step 422. If the check indicates that the process is not completed, flow continues to step 404.

If the end of data is reached but the completion flag "Done" is still false, flow continues to a set done step 418. At step 418, the completion flag "Done" is set to logical true, and flow continues to decision 412 where the last run-length will be output and flow will eventually exit through step 414, decision 416, decision 422, and then terminate at encode exit 428.

It is possible for the repeat count C to become larger than CMAX requiring more bits than allocated by this method. This situation is handled by making the check count overflow 412 decision and executing the counter overflow step 420. At step 420, the counter C is masked and bit-wise OR-ed with P shifted left by U bit positions and is placed in the A at the next available location as indexed by the encoded length L. Then, continuing inside flowchart step 414, L is incremented, the repeat count C is decrement by CMAX. After step 420, flows continues to the check count overflow 412 decision. Thus when the

encode value E repeats more that CMAX times, multiple sets of repeat counts and encoded values are output to the encoded data 140 buffer.

This entire process is repeated for each image or video frame selected during optional image sub-sampling (see 110 in Fig 1) and the encoded length L is transmitted with the encoded data associated with each frame. The encoded length varies from frame to frame depending on the content of the image being encoded.

Fig 4B–Image and Pixel Stream

Fig 4B illustrates an image and its corresponding stream of pixels. A rectangular image 430 is composed of rows and columns of pixels. The image 430 has a width 440 and a height 450, both measured in pixels. In this illustrative embodiment, pixels in a row are accessed from left to right. Rows are accessed from top to bottom. Some pixels in the image are labeled from A to Z. Pixel A is the first pixel and pixel Z is the last pixel. Scanning left to right and top to bottom will produce a pixel stream 460. In the pixel stream 460, pixels A and B are adjacent. Also pixels N and O are adjacent even though they appear on different rows in the image. If adjacent pixels have the same code the process in Fig 4A will consider them in the same run.

Because the video signal being digitized is analog there will be some loss of information in the analog to digital conversion. The video digitizing hardware can be configured to sample the analog data into the image 430 with almost any width 440 and any height 450. The present invention achieves most of its effective compression by sub-sampling the data image with the width 440 value less than the conventional 640 and the height 450 value less than the convention 480. In a preferred embodiment of the invention, for use in a medical application with T1 Internet transmission bandwidth, image dimensions are sub-sampled at 320 by 240. However a image dimension sub-sampling resolution of 80 by 60 may be suitable for some video application.

Figs 5A to 5C–Run-length Encoding Formats of the RHN Method

Figs 5A to 5C show use of a different structure than the present invention. Figs 5A to 5C show the formats for the run-length encoding of RHN. In Fig 5A, a code byte 500, with its high order bit designated as a flag bit 510.

Fig 5B shows a repeat code 520 comprising a Boolean value one in its flag bit 510 and a 7 bit count 530 in the remaining 7 low order bits. The seven bit count 530 can

represent 128 values with a zero representing "128" and 1 through 127 being their own value.

Fig 5C shows a data code 550 comprising:

1. a Boolean value zero in its flag bit 510
2. two unused data bits: data bit 6 reference by 565 and data bit 5 reference by 570, and
3. five bits, data bits 4 to 0, reference by 575, 580, 585, 590, and 595, respectively.

Fig 5C shows that in every byte of the RHN data code 550 two bits are unused and one bit is used for the flag bit, so that only five of the eight bits are used for data. The remaining three bits are wasted bits 560. The present invention uses a different structure by placing the repeat count in bits that the RHN format would not have used for data (U). The corresponding ZLN format, ZL5 (where N is 5, U is 3, and W is 8), always uses five bits for data and the remaining 3 bits for the repeat count. In practice, repeat counts are small and often can fit in 3 bits, so this embodiment of the present invention will result in superior compression performance over the RHN method.

In addition, the present invention provides for a larger count when the bit filtering is larger. For example, the alternate ZLN format where each byte contains 4 data bits, ZL4 (where N is 4 and U is 4), allows for a four bits of repeat count. For example, in practice, ZL4 is superior to RHN on a typical ultrasound image containing 16 shades of gray.

Fig 6–Encoded Data Stream

Fig 6 shows a series of decimal values 610 comprising a first value 620 equal to decimal 0, a second value 622 equal to 0, a third value 624 equal to 0, a fourth value 626 equal to 0, a fifth value 628 equal to 0, a sixth value 630 equal to 2, and a seventh value 632 equal to 10. After the encoding step 130 (Fig 1), the corresponding encoded data 140 (Fig 1) would be compressed down to three bytes of binary code 640 comprising a first byte 650, a second byte 653, and a third byte 656 each containing a merged value and count, (651, 652), (654, 655), and (657, 658), respectively. The first data 651 has a binary value of "00000" which equals the repeated decimal value zero. The first count 652 has a binary value "101" which equals decimal five representing the run-length of the repeating value in the first five of the decimal values 610. The second data 654 has a binary value of "00010"

which equals the non-repeated decimal value two. The second count 655 has a value of 1. The third data 657 has a binary value of "01010" which equals the non-repeated decimal value ten. The third count 658 has a value of 1.

Fig 7–RHN Codes and Encoded Stream

Fig 7 shows the same series of decimal values 610 (Fig 6) comprising the first value 620 equal to decimal 0, the second value 622 equal to 0, the third value 624 equal to 0, the fourth value 626 equal to 0, the fifth value 728 equal to 0, the sixth value 730 equal to 2, and the seventh value 732 equal to 10. After encoding by RHN, the corresponding encoded data 140 (Fig 1) would be compressed down to four bytes of RHN binary code 740.

The embodiment of the present invention shown in Fig 6 only requires three bytes to encode the same data. In this example, the present invention is 25% better than the RHN format.

Figs 8A and 8B–ZLN Formats

The ZLN method of the present invention provides for variable formats. The values of N 300, U 301, and W 302 can by dynamically changed between frames. For ease of communication a format is named with the prefix "ZL" and a digit representing the value of N. For example, "ZL5" refers to a format where bit width of N is equal to 5. There are multiple values of U depending of the W. To also specify the bit width of U a hyphen and a number can be appended. For example, "ZL5-13" represents a format where N = 5 and U = 13. "ZL5-3"is a common format and may be imprecisely referred to as "ZL5."

Figure 8A shows a number of formats with adjacent labels: ZL3 803, ZL4 804, ZL5 805, ZL8 808, ZL9 809, and ZL12 812. Data bits are represented by "D," and count bits are represented by "C".

Figure 8B shows how the most significant 3 bits of each color component (216, 214, and 212 of Fig 2B) are extracted and formatted in ZL9-7C format (the "C" append indicates that the color is preserved). With three red bits represented by "R", three green bits represented "G" and three blue bits represented by "B".

Decoding

To decode the compressed array, the decoder has a decode table that corresponds with the encode table. For W*4 bit color pixels, the decode table contains the appropriate alpha, red, green, and blue values. For W*3 bit color pixels, the alpha value is not used. The

compressed array is processed W bits at a time as X. The repeat count, C, is extracted from X by masking off the data value (C = X & (((2**N)-1) << U)). The encoded value, E, is extracted from X by masking off the count (E = X & ((2**U)-1)). The encoded value, E maybe used to index into the decryption. The decoded pixels are placed in a reconstructed image and repeated C times. Each element of the compressed array, A, is processed until its entire length, L, has been processed.

Fig 9–Decode Flowchart

Fig 9 illustrates the decode flowchart which presents the details of the decryption embodiment of the decode step 160 (Fig 1) and the image reconstitution step 180 (Fig 1).

The decoding beings at a decode entry 900. In a "decode initialization" step 901, a repeat counter C is set to one, an encoded length L is set to the value obtained with the encoded data 140 (Fig 1), and an index I is set to 0. Next, a "get code" step 902 obtains a signed byte X from the encoded data 140 (Fig 1) array A. The index I is incremented. The count (for example the 3-bit count 380 as shown in Fig 3B) is extracted from X by masking off the data bits and placed in the repeat counter C (C= X & ((2**N)-1 << U). The value of E is extracted from X by masking off the count bits (E = X & (2**U)-1). In practice, the count mask and value mask can be pre-computed with the following two lines of code in the C programming language:

```
valueMask = -1 << U;
countMask = ~valueMask;
```

In this illustrative decryption embodiment of the present invention, flow goes to a "decode lookup" step 908 where the value of E is used to index into the decode table 1110 (Fig 11) to obtain a pixel value V. In the other embodiments where E is not encrypted, E is used as V and step 908 is bypassed. Flow continues to a "check zero count" 909 decision.

The 909 decision always fails the first time ensuring that a place pixel step 910 is executed. The place pixel step 910 places the pixel value V in the next location of the decompressed image and decrements the repeat counter C and returns to the 909 decision. The pixel value V is placed repeatedly until C decrements to zero. Then the 909 decision branches flow to a "reset counter" step 914. At step 914 the repeat counter is reset to 1.

Flow continues to the "check length" 916 decision where the index I is compared to the encoded length L to determine if there are more codes to be processed. If I is less than L flow returns to step 902, otherwise the decode process terminates at a "decode exit" 918.

The entire decode process is repeated for each encoded frame image.

Fig 10–Interpolation

Fig 10 interpolation when a two adjacent pixels 1010 and 1012 and two subsequent row adjacent pixels 1014 and 1016 are stretched to insert a new row and column of pixels.

Pixels 1052, 1054, 1056, 1058 and 1060 are inserted due to the enlargement of the image. Their values are calculated by averaging the values of the two pixels above and below or to the left or the right of the new pixel. A preferred sequence is calculation of:

1. 1052 between 1010 and 1012
2. 1054 between 1010 and 1014
3. 1058 between 1012 and 1016
4. 1056 between 1054 and 1058

Pixel 1060 can be calculated on the interpolation for the subsequent row.

Fig 11–Encryption

By using corresponding encoding and decoding tables the data can be encrypted and decrypted without using actual values. Encryption provides a level of security for the encoded data 140 while in storage or transit.

Fig 11 shows an example of an encryption table 1100, where N is 3 and W is 8, and a decryption table 1110, where N is 3 and U is 5.

The encode table 1100 is 2 the power of N in length. If the target color image format is W*4 bit color, then the decode table 1110 has W bits for alpha, red, green, and blue each, respectively. If the target color image format is W*3 bit color, then the alpha value is not used. If the image is W bit grayscale then only the grayscale value is used to create the decompressed and decoded image.

The corresponding table elements are mapped to each other. For example, 0 could encode to 22 as long as the 22nd element of the decode table returns (`0xff << 24 | 0 << 16 | 0 << 8 | 0`).

When these versions of the tables are used, the encode and decode processes and their speed of execution are substantially the same but the encoded data 140 (Fig 1) becomes a cipher and has a higher level of security. It should be recognized by one with ordinarily

skill in the art that there are other embodiments of the present invention with different encryption/decryption table rearrangements.

Advantages

Noise Filtering and Image Enhancement

The removal of the least significant bits of pixel values results in high quality decompressed images when the original image is generated by an electronic sensing device such as an ultrasound machine which is generating only a certain number of bits of grayscale resolution. By variably altering the number of most significant bits various filters can be implemented to enhance the image quality. Such a noise filter can be beneficial when the image is generated by an imaging technology such as radar, ultrasound, x-ray, magnetic resonance, or similar technology. Variations can be made to enhance the perceived quality of the decompressed image. Therefore, altering the number of data bits selected and altering the width of the repeat count is anticipated by this invention and specific values in the examples should not be construed as limiting the scope of this invention.

Dynamic Variable Formats

While a video stream is being viewed a viewer on the decoding end of the transmission can vary the settings for the compressor. Different tradeoffs between image spatial and temporal quality can be made. As the contents of the video signal change an appropriate format can be selected. Control signals can be sent back to the compressor via a communications link.

Execution Speed

The preferred embodiment of this invention use a number of techniques to reduce the time required to compress and decompress the data.

The methods require only a single sequential pass through the data. Both the compression steps 100 and the decompression steps 150 access a pixel once and perform all calculations.

When selecting the filtered pixel value 299, the preferred embodiment selects the low order byte from the 32 bit pixel value 200 or the 24 bit pixel value 210 so that an additional shift operation or addressing operation is avoided.

The shift operation is a fast and efficient way to convert a byte or word to the filtered pixel value 299.

General Purpose

The lossless compression of the sampled data achieved by the preferred embodiment of the present invention results in high quality video streams that have general purpose application in a number of areas including, without limitation, video conferencing, surveillance, manufacturing, rich media advertising, and other forms of video transmission, storage, and processing.

Lossless Nature/No Artifacts

Once the analog signal is sub-sampled and filtered to select a filtered pixel value which eliminates some of the real world defects, the methods of the present invention compress and decompress the data with no irreversible data loss. Unlike JPEG and MPEG, the decompressed image never suffers from artificially induced blocking or smearing or other artifacts that are result of the lossy compression algorithm itself. As a result even a small sub-sample of the image remains clear and true to the perceived quality of the original image.

Superior Features over RHN Format

When compared against the RHN format, the format and methods of the present invention provide a number of advantages, including, but not limited to, faster speed and smaller size of encoded data, better performance for both medical and typical video images, and a typically closer representation of the original video signal.

Conclusion, Ramification, and Scope

Accordingly, the reader will see that the compression and decompression steps of the present invention provides a means of digitally compressing a video signal in real time, communicating the encoded data stream over a transmission channel, and decoding each frame and displaying the decompressed video frames in real time.

Furthermore, the present invention has additional advantages in that:

1. it provides a means of filtering real world defects from the video image and enhancing the image quality;

2. it allows for execution of both the compression and decompression steps using software running on commonly available computers without special compression or decompression hardware;
3. it provides decompressed images that have high spatial quality that are not distorted by artifacts of the compression algorithms being used;
4. it provides a variably scalable means of video compression; and
5. it provides a means for reducing the space required in a storage medium.

Although the descriptions above contain many specifics, these should not be construed as limiting the scope of the invention but as merely providing illustrations of some of the preferred embodiments of this invention. For example, bit ordering can be altered and the same relative operation, relative performance, and relative perceived image quality will result. Also, these processes can each be implemented as a hardware apparatus that will improve the performance significantly.

Thus the scope of the invention should be determined by the appended claims and their legal equivalents, and not solely by the examples given.

Claims: I claim:

1. (Amended) A method of compression of graphic images which make up a video stream, comprising the steps of:
   (a) sub-sampling a number of pixel bits from an image selected from said graphic images;
   (b) run-length encoding repeated instances of said number of pixel bits;
   repeating steps (a) and (b) until each said number of pixel bits is encoded in an encoded data buffer.
2. The method of claim 1 wherein the image dimensions are less than or equal to 320 by 240.
3. The method of claim 1 wherein said number of pixel bits is one of the set of 3, 4, 5, 8, 9, 12, 15, 16, and 24.
4. The method of claim 3 wherein said number of pixel bits is extracted from the most significant bits of each color component.
5. (Amended) An encoded video signal comprising a series of said encoded data buffers, wherein said data buffers were encoded according to the method of claim 1.
6. A storage medium in which the encoded video signal as claimed in claim 5 is stored.
7. (Amended) A method of decompressing an encoded video signal, comprising the steps of:
   (a) reading a stream of run-length encoded codes;
   (b) determining a series of pixels based on the values and run-lengths of said codes;
   (c) combining said pixels into an image.
8. The method of claim 7 further comprising the step of displaying a series of said images.
9. The method of claim 7 wherein the width and the height of said image are less than or equal to 320 by 240, respectively.
10. The method of claim 7 wherein said codes represent the combination most significant bits of each of the color components of each pixel.
11. (Amended) A machine for compressing of a plurality of video frames which make up a video signal, comprising:

(a) a video digitizer configured to digitizing a frame from said video frames;

(b) a video memory which is able to receive a plurality of pixels from said video digitizer;

(c) an encoding circuit for counting repeated instances of a pixel value when scanning said plurality of pixels and outputting a series of encoded data comprising a combined run-length field and a data field.

(d) a memory which is able to store said encoded data;

(e) an input/output device.

12. The machine of claim 11 wherein said encoding circuit variably selects one of a set of 3, 4, 5, 8, 9, 12, 15, 16, and 24, as the number of pixel bits.

13. The machine of claim 12 wherein said pixel value is extracted from the most significant bits of each color component.

14. The machine of claim 11 wherein said input/output device is a storage medium.

15. The machine of claim 11 wherein said input/output device is a communications transmission channel.

16. A machine for decompressing an stream of encoded data that represents a video signal, comprising:

(a) an input/output device for reading said stream of encoded data;

(b) a decoding circuit which can decode the encoded data and output a stream of pixel values; and

(c) a memory that is able to store an image comprising said stream of pixel values that can be displayed as frames of a video sequence.

17. The method of claim 1 wherein one or more of the settings of width, height, frame rate, brightness, and contrast of said images are variably altered by a receiver of said encoded data.

18. The method of claim 1 wherein said number of pixel bits are variably altered by a receiver of said encoded data.

19. The method of claim 1 further comprising a step of compressing said buffer with a lossless technique known in the art.

20. The method of claim 8 wherein said images are enlarged by stretching prior to said displaying.

21. The method of claim 1 further comprising a step of encrypting said number of pixel bits.

22. The method of claim 1 wherein said graphic images have a first predetermined frame rate and a subset of said graphic images are sub-sampled at a second frame rate that was less than the first frame rate such that only a subset of said graphic images are selected from the original set of said graphic images, and
wherein said image selected from said graphic images is a sub-sampled image such that it is one of said subset of sub-sampled images.

23. The method of claim 1 wherein the image dimensions of said video stream is greater than 320 pixels wide and 240 pixels high, and
wherein said method further comprises the step of first dimensionally sub-sampling an image from said graphic images such that the sub-sampled image dimensions of said image are less than or equal to 320 by 240.

24. The method of claim 1 wherein a length of the encoded data in said encoded data buffer in placed in said encoded data buffer.

25. The method of claim 7 further comprising the step of reading a length of the encoded data and using said length to determine when all the encoded data has been processed.

30

VARIABLY GENERAL PURPOSE COMPRESSION FOR VIDEO IMAGES (ZLN)

Abstract: Methods, medium, and machines which compress, enhance, encode, transmit, decompress and display digital video images in real time. Real time compression is achieved by sub-sampling each frame of a video signal, filtering the pixel values, and encoding. Real time transmission is achieved due to high levels of effective compression. Real time decompression is achieved by decoding and decompressing the encoded data to display high quality images. Receiver can alter various setting including but not limited to the format for the compression, image size, frame rate, brightness and contrast.

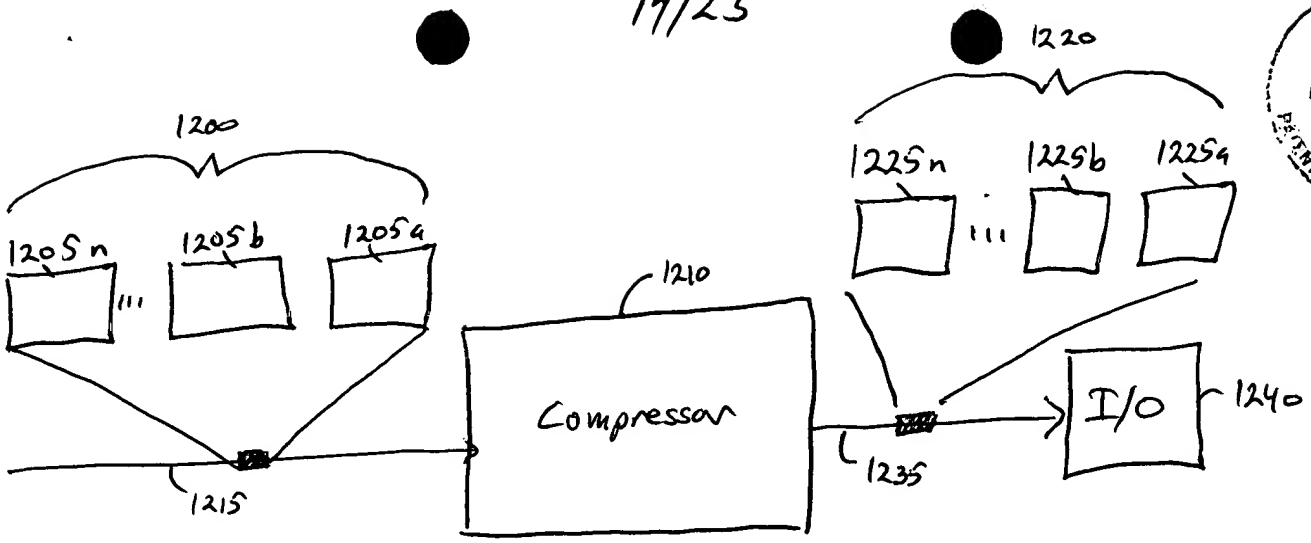


Fig 12A

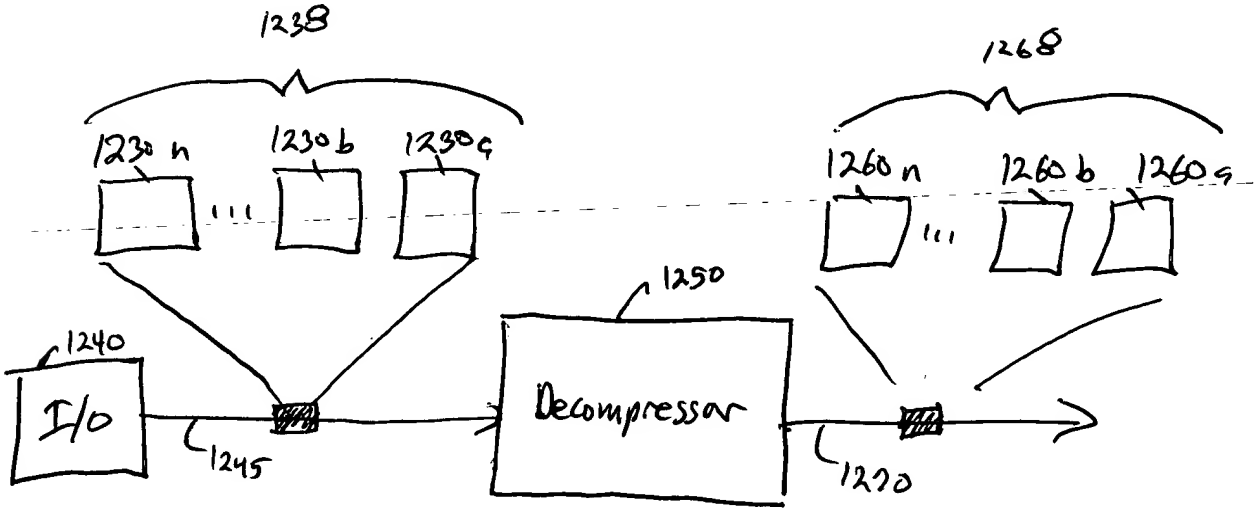


Fig 12B

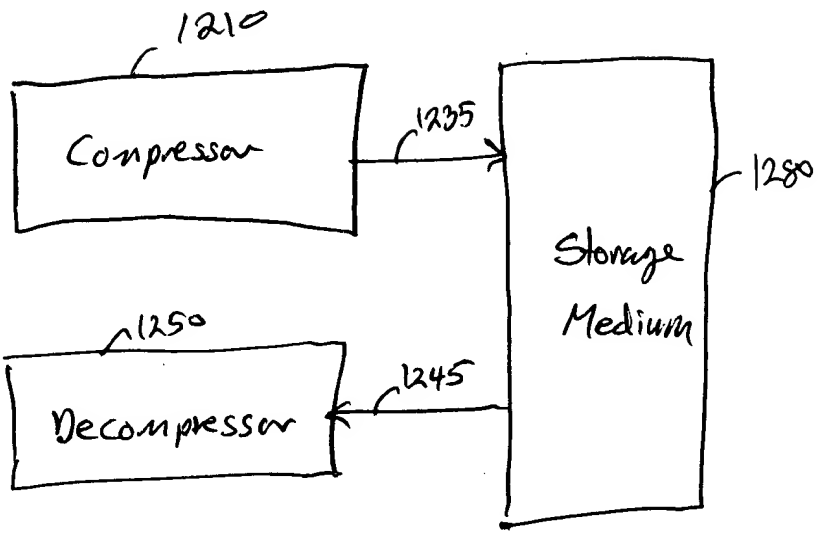


Fig 12C

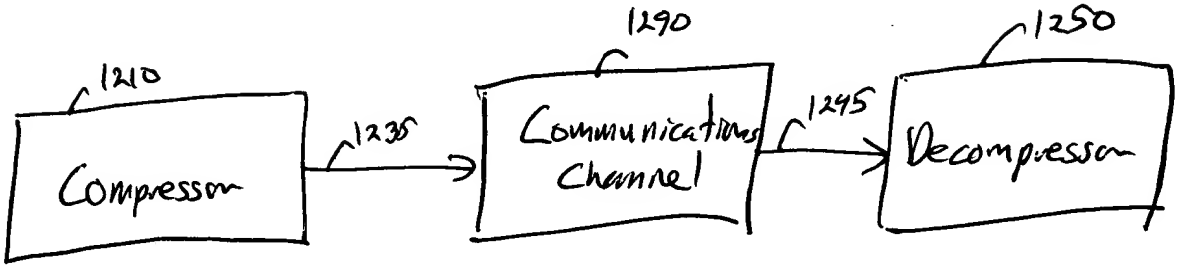


Fig 12D

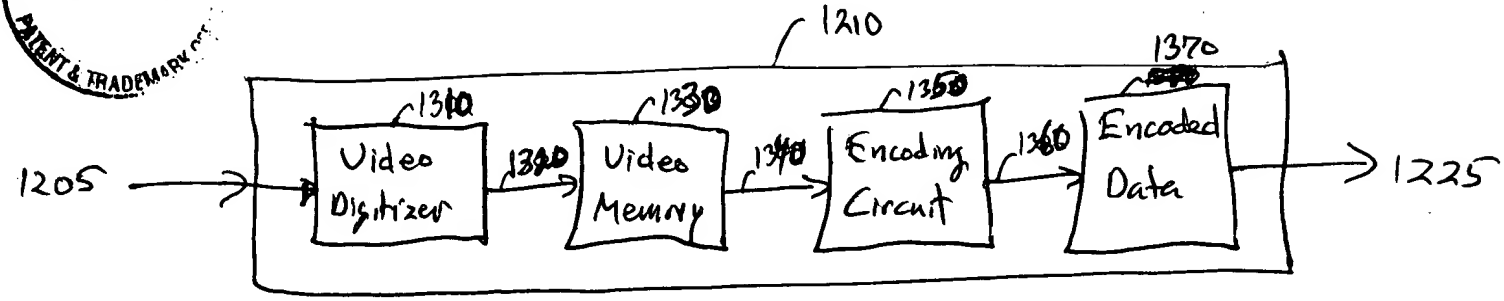


Fig 13A

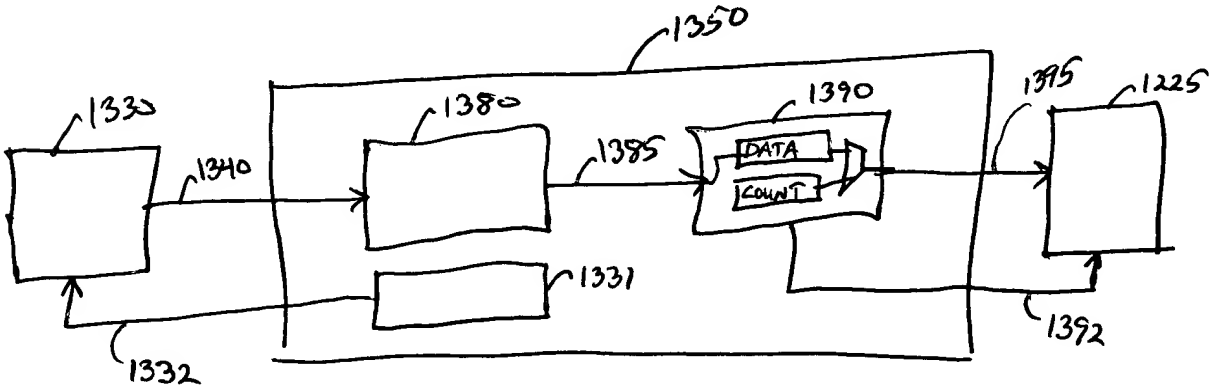


Fig 13B

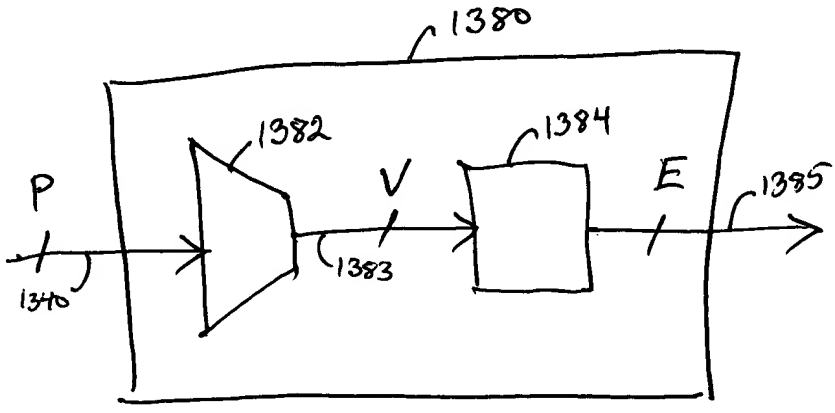


Fig 13C

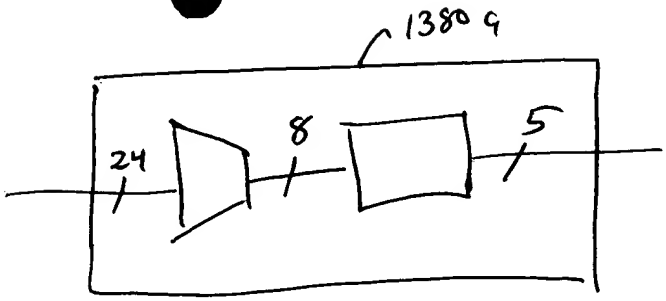


Fig 13D

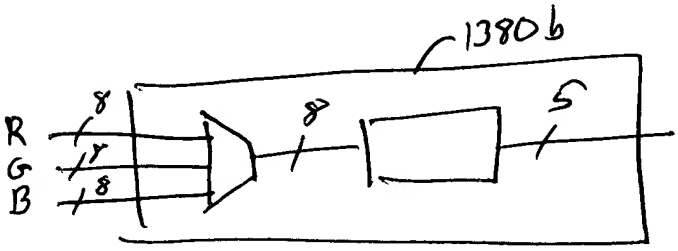


Fig 13E

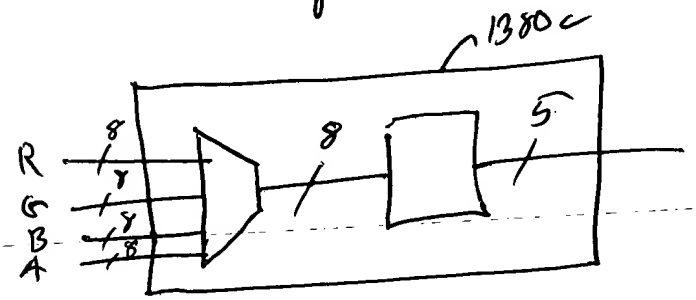


Fig 13F

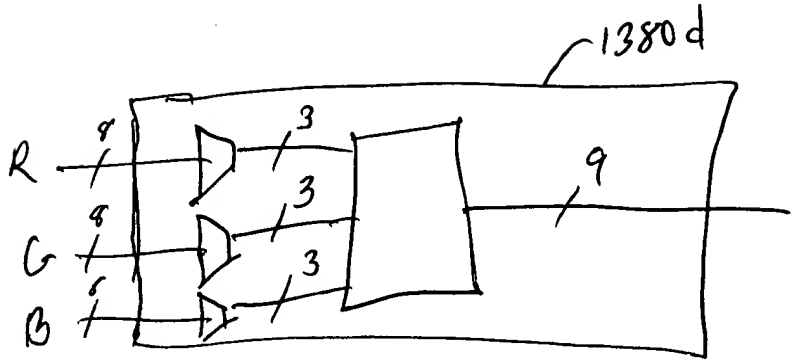


Fig 13G

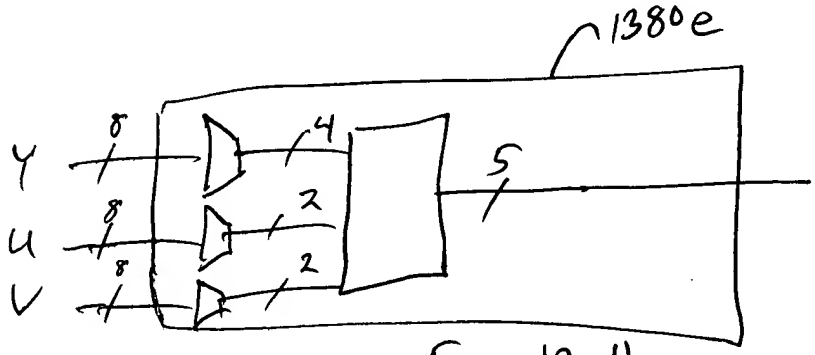


Fig 13H

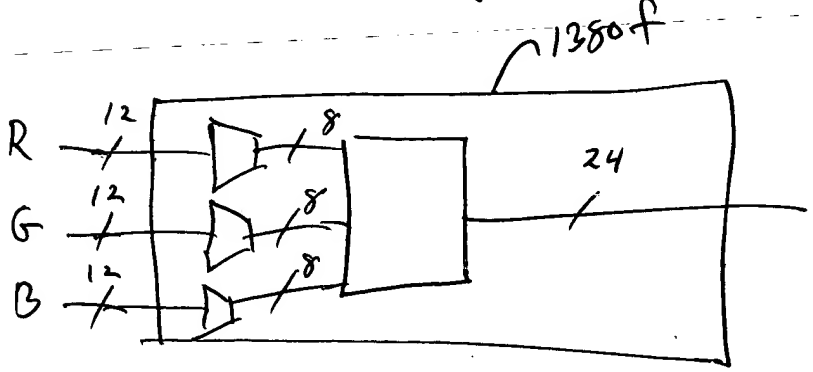


Fig 13I

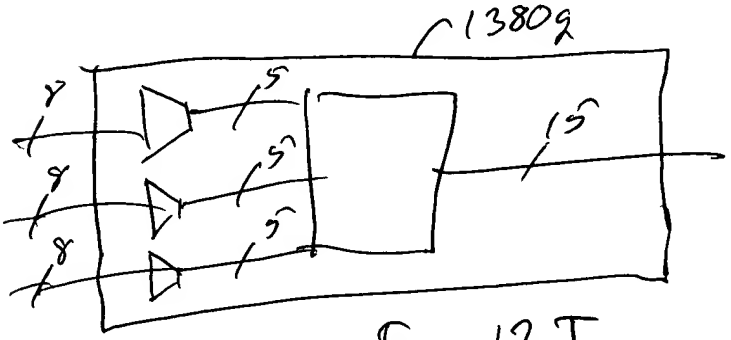


Fig 13J

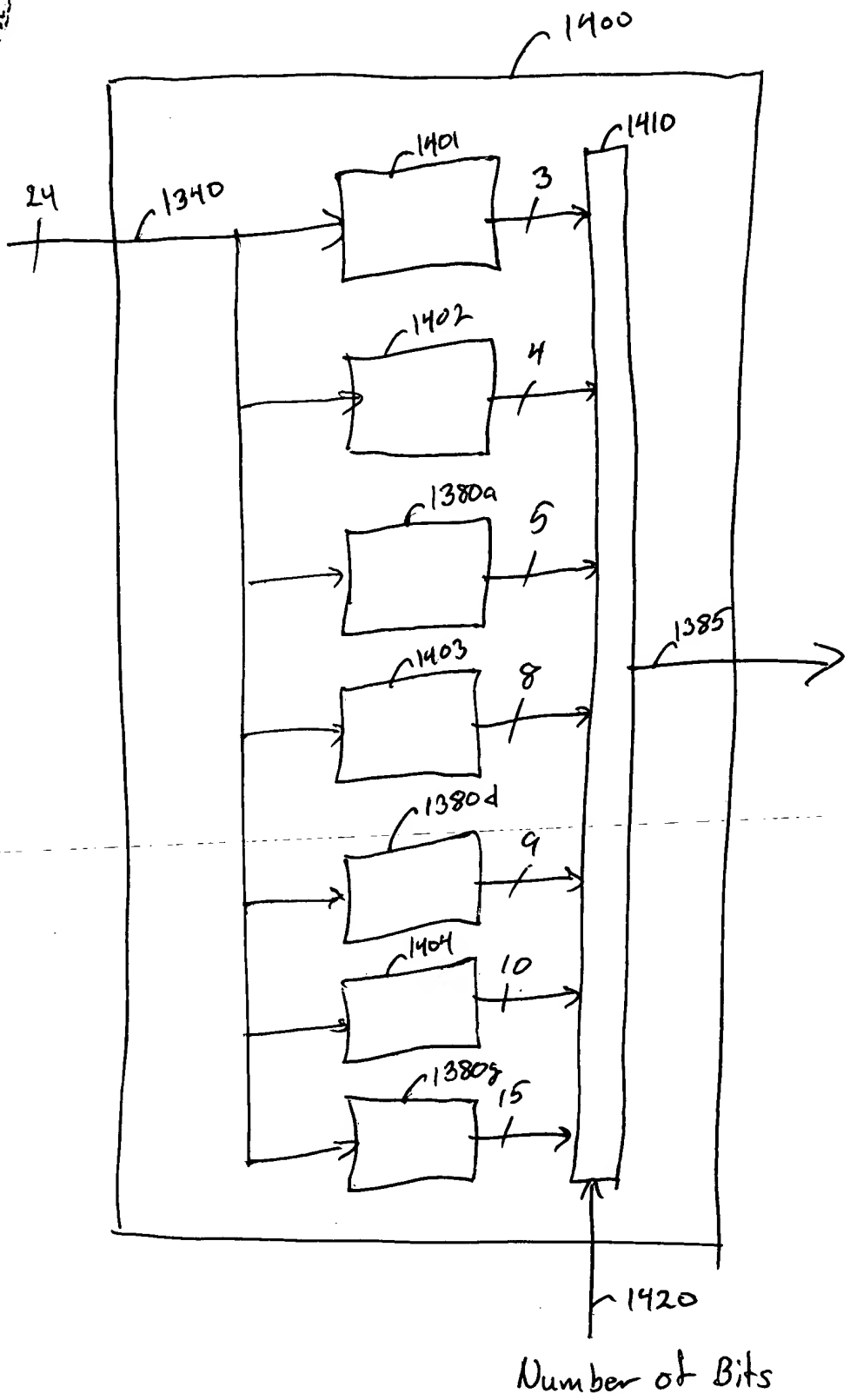


Fig 14A

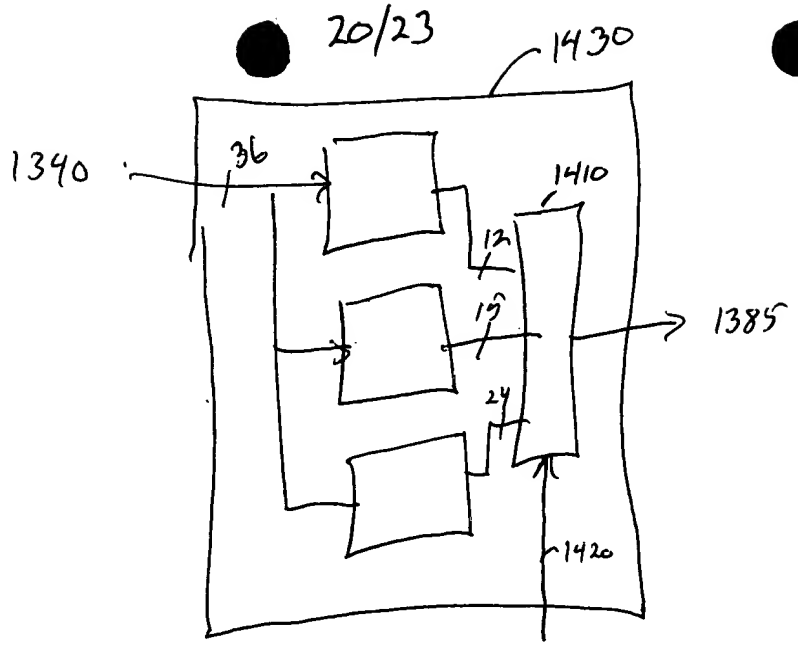


Fig 14 B

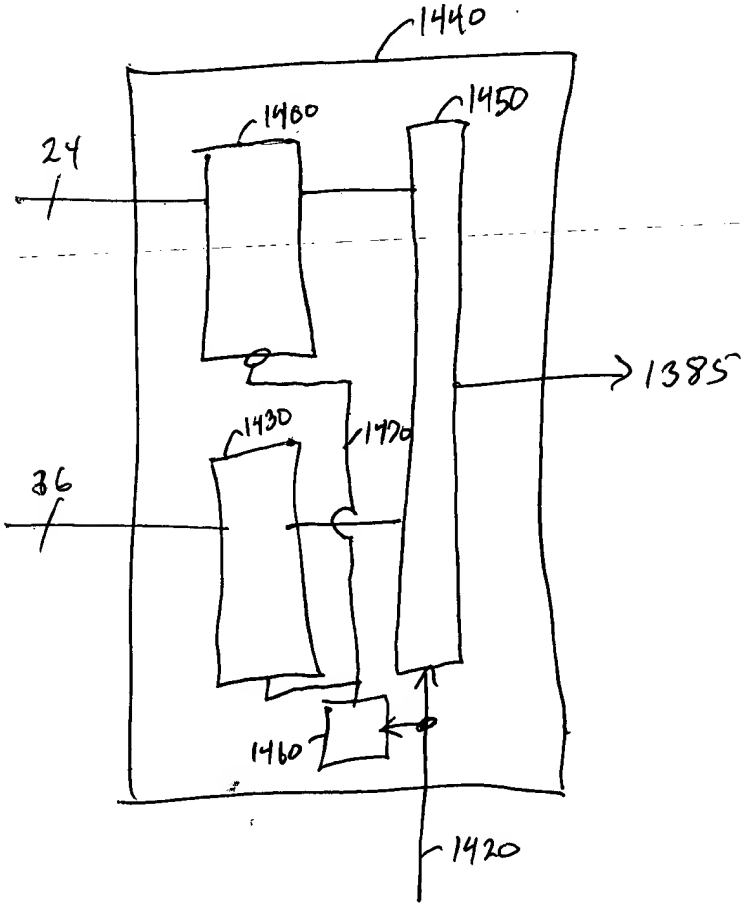


Fig 14 C

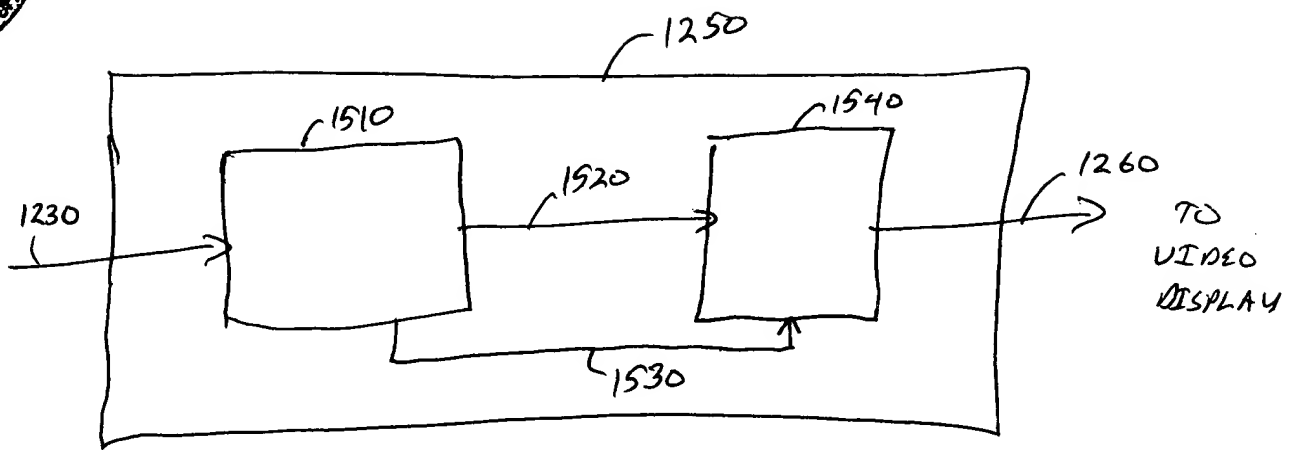


Fig 15

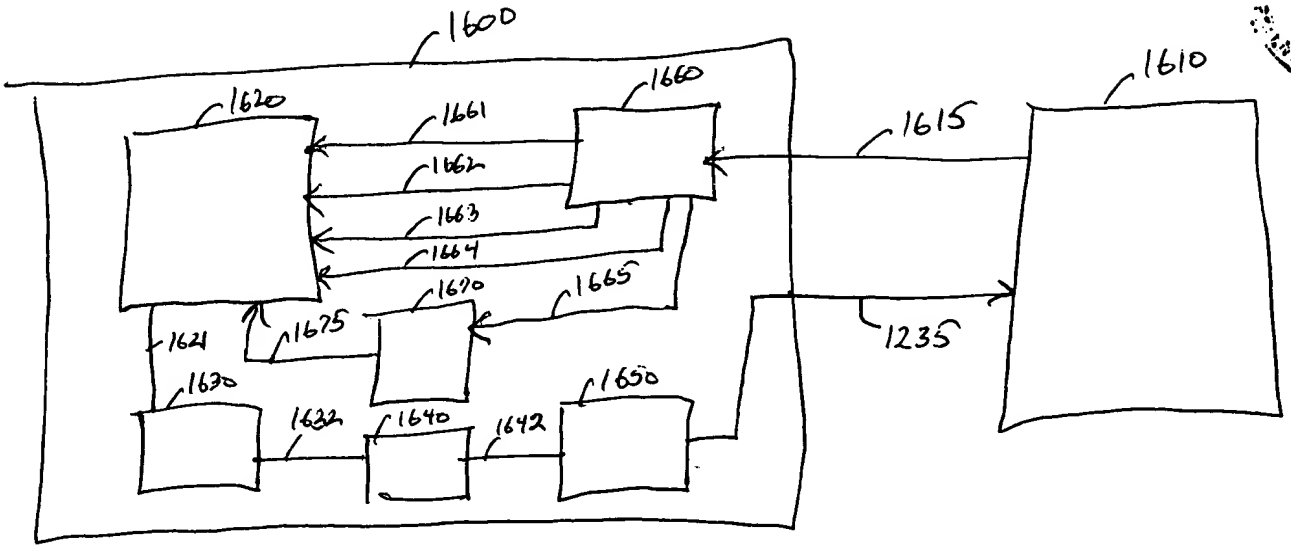


Fig 16A

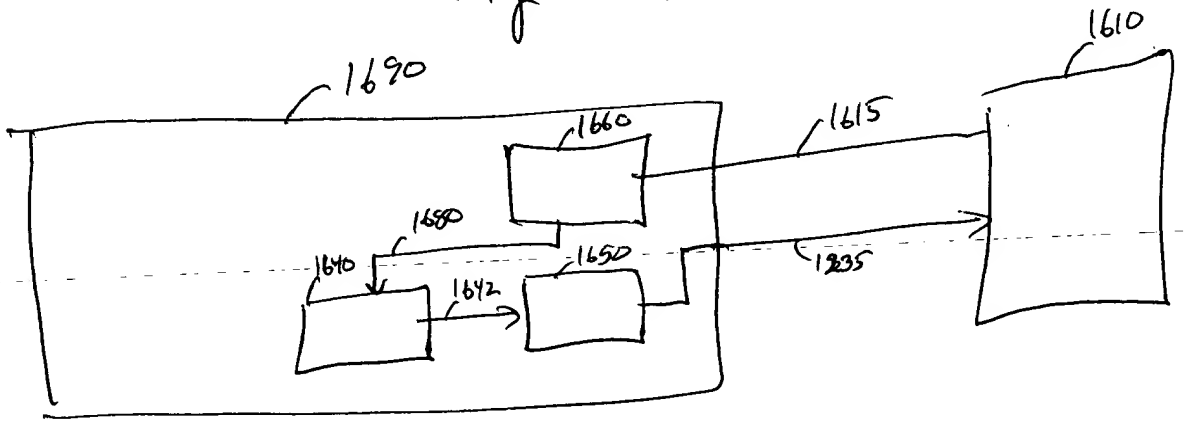


Fig 16B

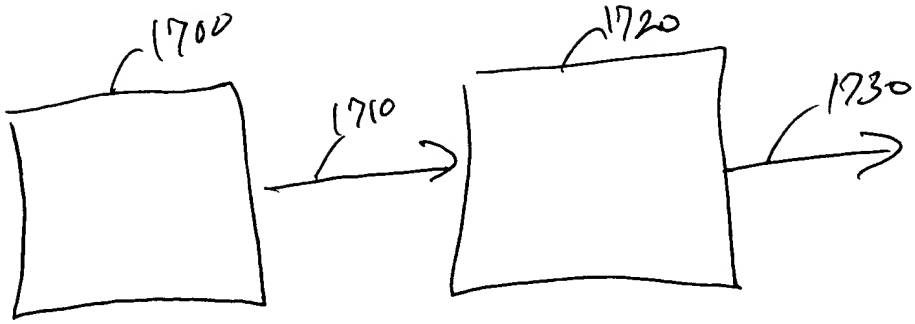


Fig 17

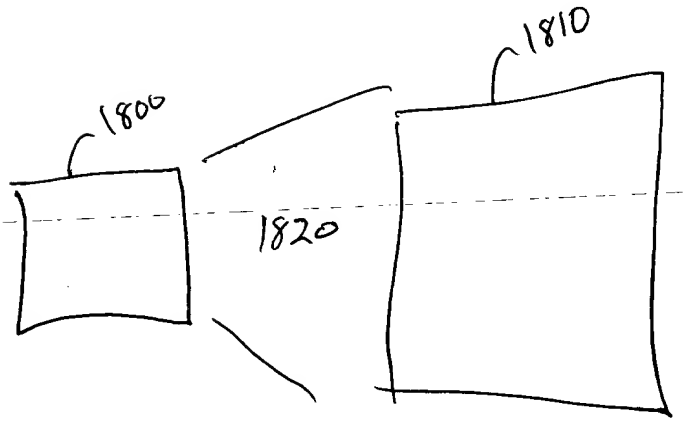


Fig 18